Ítalo Herbert Santos e Gomes

TVOICE: UM SISTEMA DE MANIPULAÇÃO DE LINGUAGENS PARA AUXILIAR PORTADORES DE NECESSIDADES ESPECIAIS ATRAVÉS DA WEB

Natal - RN
Março de 2005

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-graduação em Sistemas e Computação

TVOICE: UM SISTEMA DE MANIPULAÇÃO DE LINGUAGENS PARA AUXILIAR PORTADORES DE NECESSIDADES ESPECIAIS ATRAVÉS DA WEB

Ítalo Herbert Santos e Gomes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, na Área de Concentração de Redes e Sistemas Distribuídos.

Orientador:
Prof. Dr. Guido Lemos de Souza Filho

Co-Orientador:
Prof. Dr. David Boris Paul Déharbe

Natal - RN
Março de 2005

Dedicatória e Agradecimentos

Dedico a todos que, de alguma forma, me ajudaram em mais essa etapa de minha vida. E agradeço a Deus, a meus orientadores, aos meus pais, a minha noiva, aos meus amigos, aos meus colegas e, enfim, a todos que contribuíram e me incentivaram, possibilitando mais essa conquista.

“Confia ao Senhor as tuas obras, e os teus desígnios serão estabelecidos”.

Provérbios 16:3.

Resumo

Este trabalho apresenta o tVoice, *software* que manipulando linguagens de marcações e extraindo informações, sendo parte integrante do sistema VoiceProxy, auxilia portadores de necessidades especiais no acesso à Web.

Esse sistema é responsável pela recuperação e tratamento dos documentos na Web, realizando a extração de informações textuais contidas nos mesmos, sendo ainda, capaz de gerar ao final, através de técnicas de tradução, um script de áudio, a ser utilizado pelo subsistema de interface do VoiceProxy, o iVoice, no processo de síntese de voz dessas informações para portadores de deficiência visual.

Nesta etapa o tVoice, além do tratamento da linguagem de marcação HTML, processa outros dois formatos de documentos, o PDF e o XHTML.

Para permitir que, além do iVoice, outros subsistemas de interface possam fazer uso do tVoice, através de acesso remoto, são utilizadas técnicas de distribuição de sistemas, que baseadas no modelo cliente-servidor proporcionam um funcionamento assemelhado a um servidor proxy de tratamento de documentos.

Palavras-chave: Sistemas Distribuídos, Cliente-Servidor, Tradutores, Acessibilidade.

Abstract

This work presents the tVoice, software that manipulates tags languages, extracting information and, being integral part of the VoiceProxy system, it aids bearers of special needs in the access to the Web.

This system is responsible for the search and treatment of the documents in the Web, extracting the textual information contained in those documents and preceding the capability of generating eventually through translation techniques, an audio script, used by the of interface subsystem of VoiceProxy, the iVoice, in the process of voice synthesis.

In this stage the tVoice, besides the treatment of the tag language HTML, processes other two formats of documents, PDF and XHTML.

Additionally to allow that, besides the iVoice, other interface subsystems can make use of the tVoice through remote access, we propose distribution systems techniques based in the model Client-Server providers operations of the fashion of a proxy server treatment of documents.

Keywords: Distributed Systems, Client-Server, Translators, Accessibility.

Sumário

Lista de Figuras

Lista de Tabelas e Quadros

TABELAS

QUADROS

1. Introdução

Desde a idealização da Internet como meio de propagação de informações e interligação de fronteiras, é crescente o número de sistemas que utilizam tal estrutura e suas funcionalidades, explorando o potencial uso dos mais variados recursos disponibilizados nesse ambiente.

Atualmente, através da integração de diferentes redes físicas e lógicas, navegando-se através de diversos protocolos e formas de comunicação, encontramos inúmeras aplicações que de forma distribuída, dispersas em diferentes máquinas espalhadas pela rede, trabalham em conjunto, transparecendo para os usuários apenas o funcionamento de um único sistema.

Para que possam trabalhar em conjunto e assim solucionar problemas, as máquinas e seus sistemas precisam utilizar tecnologias, tais como técnicas de distribuição de sistemas e conjuntos de protocolos, que viabilizem a comunicação e a organização dos eventos envolvidos, para que assim possam realizar suas tarefas de maneira ordenada.

Tais técnicas devem garantir que características encontradas em sistemas distribuídos, tais como transparência e confiabilidade, possam funcionar adequadamente e, assim, atingir os resultados desejados.

Na Internet, dentre os modelos de estruturação de distribuição de sistemas encontrados, o mais utilizado ainda é o modelo Cliente-Servidor [03,04,05].

Mesmo dentre a imensurável diversidade de aplicações que utilizam tal estrutura, encontram-se aquelas que trabalham para auxiliar pessoas no próprio acesso à Web. Nesse contexto insere-se a Acessibilidade, como um movimento que busca a igualdade de condições de acesso aos serviços de informação, documentação e comunicação para todos, fazendo uso também da distribuição de sistemas no desenvolvimento de ferramentas que possam facilitar a vida de portadores de necessidades especiais no acesso às informações disponibilizadas nesse meio.

O VoiceProxy, projeto no qual este trabalho se insere, nesta etapa, sendo ainda um sistema de apoio a deficientes visuais no acesso à Web, enquadra-se também no

contexto referenciado anteriormente, a saber a Acessibilidade, aspirando agora por novos objetivos.

1.1 Objetivos da Pesquisa

Os objetivos desta dissertação podem ser sintetizados nas seguintes assertivas:

- Propor e provar ser possível desenvolver um sistema de apoio à interface de áudio do VoiceProxy que possibilite a diversificação dos formatos de documentos manipulados para a extração de informação e permita sua utilização de forma distribuída, tornando-o mais acessível e flexível quanto ao atendimento aos usuários.
- Realizar testes de verificação do sistema quanto a algumas características tais como concorrência, confiabilidade e desempenho.
- Contribuir para a Acessibilidade dentro do âmbito da rede mundial.

1.2 Organização do Texto

O texto deste documento está organizado como apresentado a seguir.

No capítulo 2 são apresentadas, de forma sucinta, as tecnologias envolvidas na realização deste trabalho, a saber: sistemas distribuídos e compiladores.

O capítulo 3 apresenta trabalhos correlatos ao trabalho que foi desenvolvido, isto é, trabalhos em cujo contexto este se insere.

Em seguida, no capítulo 4 é encontrada a apresentação do Sistema VoiceProxy, sistema de auxílio à navegação de deficientes visuais na Internet, do qual este trabalho faz parte, como componente integrante.

A seguir, no capítulo 5 é apresentado o Subsistema tVoice, seu desenvolvimento e uma análise sobre os testes realizados com a sua execução.

Finalmente, no capítulo 6 podem ser encontradas as conclusões percebidas com a finalização desta dissertação, apresentando, ainda, perspectivas futuras para o trabalho desenvolvido.

2. *Fundamentação Teórica*

Das novas metas propostas para a etapa conseguinte do projeto VoiceProxy originou-se a demanda de novas versões de seus subsistemas formadores, onde ambos necessitariam se intercomunicar e interagir de forma concorrente e independente de suas localizações.

Para tal feito, especificamente para o desenvolvimento do trabalho apresentado nesta dissertação, foi realizado um estudo de embasamento para a construção do novo sistema de tradução do VoiceProxy, denominado de tVoice.

Dentre suas propriedades, nesta nova etapa, podem ser citadas: o recebimento e processamento múltiplo de solicitações concorrentes, uso do modelo cliente-servidor no serviço de atendimento à interface de áudio e processamento e manipulação de diferentes formatos de documentos.

Neste capítulo são apresentados, de forma geral, os conceitos das principais tecnologias envolvidas na evolução deste subsistema como trabalho.

2.1 Sistemas Distribuídos

Na literatura, ainda existem várias definições que conceituam sistemas distribuídos, o que prova que KIRNER e MENDES [02] estavam certos ao escrever que, durante anos, esse assunto ainda seria bastante discutido até que se chegasse a um consenso do que realmente seria um sistema distribuído.

Durante o processo de definição do significado do termo "sistema distribuído", evidenciou-se que vários elementos de um sistema podem ser distribuídos, tais como processadores, programas, dados e controle [02,03].

Por apresentar um conjunto de módulos, interligados fracamente através de um subsistema de comunicação de topologia arbitrária e funcionando através de um controle descentralizado, os sistemas que executam tarefas de programas de aplicações são assim denominados de distribuídos.

Tais sistemas são distinguidos e caracterizados por um fraco acoplamento, onde existem recursos remotos e locais, porém tal distribuição é transparente para os seus usuários [01,02].

De acordo com os escritos de KIRNER e MENDES, a distribuição, observada como uma espécie de organização, deve ser enfocada sob dois aspectos: do ponto de vista físico e do ponto de vista lógico. Fisicamente, a distribuição existirá se o sistema for composto de pelo menos dois componentes físicos autônomos. Logicamente, um sistema distribuído deve conter dois tipos de componentes: os ativos, encarregados da parte de processamento; e os passivos, componentes que contêm dados.

Observando os dois pontos de vista da distribuição, podem ser destacados, dentre os aspectos gerais de projeto encontrados em sistemas distribuídos, os seguintes [01,02,04]:

- **TRANSPARÊNCIA:** O aspecto Transparência de um sistema distribuído refere-se à capacidade do sistema em desenvolver, na mente do usuário, a imagem ou impressão de se estar trabalhando em um sistema centralizado. O conceito de transparência, dentro do escopo de sistemas distribuídos, pode ser aplicado, também, a diferentes aspectos, dentre os quais podem ser destacados:
    - **Localização:** Tal aspecto está relacionado ao fato dos usuários fazerem uso do sistema e de seus recursos sem tomarem conhecimento de sua localização física.
    - **Concorrência:** Tal aspecto está relacionado ao uso simultâneo (concorrente) dos recursos disponíveis, de forma transparente ao usuário.
    - **Paralelismo:** Tal aspecto está relacionado à subdivisão de atividades dentro do sistema, sendo estas executadas paralelamente, sem que o usuário tome conhecimento.
- **FLEXIBILIDADE:** O aspecto Flexibilidade de um sistema distribuído refere-se à capacidade de fácil modificação, inserção e remoção dos módulos que o compõem.

- **CONFIABILIDADE:** O aspecto Confiabilidade de um sistema distribuído refere-se à capacidade de garantir a disponibilidade de serviço do sistema, utilizando-se de mecanismos de tolerância àfalhas.
- **DESEMPENHO:** O aspecto Desempenho de um sistema distribuído refere-se à capacidade do sistema em minimizar a utilização dos recursos que o compõem e maximizar a sua produção.

No conceito de distribuição em sistemas existem ainda outros fatores relacionados, que fazem parte de sua caracterização, um deles é a comunicação.

Quando há a distribuição em um sistema, a comunicação se dá através da troca de mensagens, através de uma rede de interconexão, sobre um conjunto de diretivas chamadas de protocolo. Um protocolo é o conjunto de regras, padrões e especificações técnicas que regulam o fluxo de dados por meio de programas específicos [01,05].

A troca de informação entre os componentes de um sistema pode ser realizada utilizando-se dois modos de transferência [01,02,05,06]:

- **SÍNCRONO:** Neste modo de troca de mensagens o emissor e o receptor devem participar da comunicação no mesmo instante de tempo. Ou seja, há um sincronismo entre origem e destino. Tanto o emissor quanto o receptor devem estar prontos e aptos para receber e enviar mensagens. Exemplos práticos desse modo de comunicação são os utilizados pelos sistemas de troca de mensagens instantâneas, bate-papos e tele-conferência. Ver ilustração da Figura 1-a.
- **ASSÍNCRONO:** Neste modo de troca de mensagens o emissor e o receptor não são obrigados a estar participando da comunicação no mesmo instante de tempo. Ou seja, não há a necessidade de um sincronismo entre ambos. Porém, nesta forma de comunicação, deve haver tanto no receptor quanto no emissor um repositório para o armazenamento das mensagens trocadas. Exemplos práticos desse modo podem ser constantemente encontrados nos sistemas de gerenciamento de e-mails e de distribuição de notícias e propagandas. Ver ilustração da Figura 1-b.

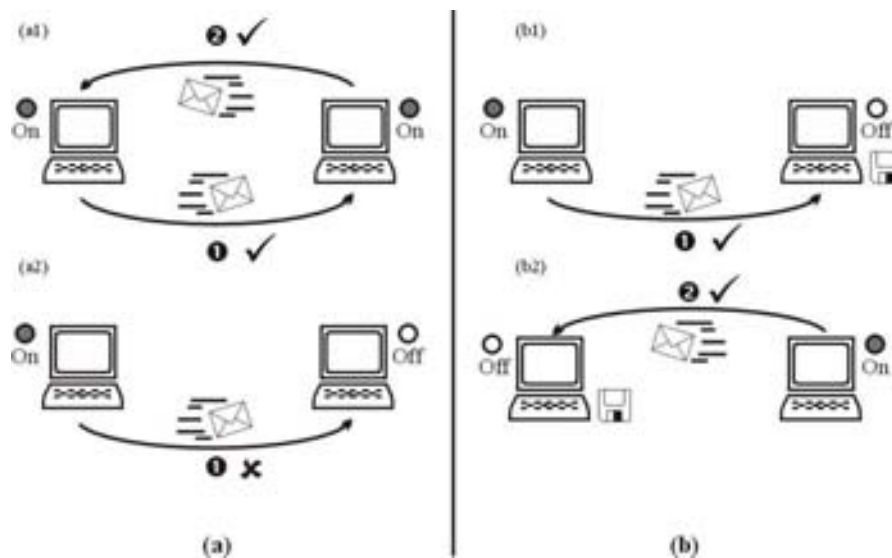



Figura 1 (a) Forma de comunicação síncrona
(b) Forma de comunicação assíncrona

Outra característica dos sistemas distribuídos é o modelo estrutural utilizado. Dentre os modelos encontrados podem ser destacados [01,02,04]:

- **MODELO DE PARES:** Neste modelo, os componentes são semelhantes e dispostos em pares, comunicando-se entre si. Nele, todos os componentes realizam o mesmo tipo de tarefa. Este modelo é amplamente utilizado em sistemas de processamento paralelo, cujo objetivo é obter ganhos na velocidade da solução. Podem ser citados como exemplos clássicos os sistemas de multiplicação de matrizes. Ver modelo na Figura 2-a.
- **MODELO DE FILTROS:** Neste modelo, os componentes são organizados em camadas ou níveis, de maneira que as respostas dos componentes da camada anterior sirvam de entrada para os componentes da camada imediatamente seguinte. Os componentes são distribuídos de acordo com a funcionalidade desempenhada. Como exemplo, temos os *Pipelines*[1] funcionais e os sistemas de filtragem de sinais. Ver modelos nas Figuras 2-b.
- **MODELO CLIENTE-SERVIDOR:** Neste modelo de sistema, os componentes são classificados em dois tipos: Clientes e Servidores.

---

[1] **Pipelines**. Métodos ou funções cujas saídas servem de entradas para outros métodos ou funções.

- **Clientes:** São os componentes que solicitam serviços aos Servidores.
- **Servidores:** São os componentes que oferecem serviços aos Clientes.

Exemplos prático e largamente encontrado no âmbito da Internet são: os *browsers*, como clientes, e os servidores Web, como prestadores do serviço de entrega de documentos.

(a) Modelo de Pares	(b) Modelo de Filtros	(c) Modelo Cliente-Servidor

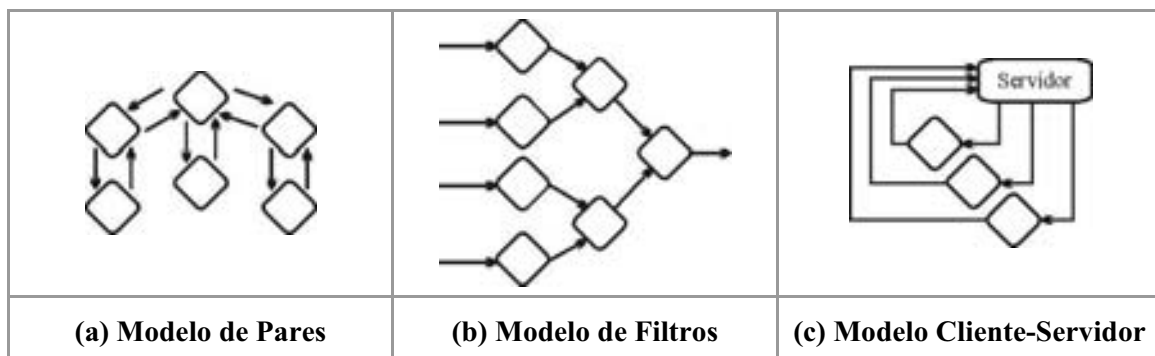



Figura 2 Modelos estruturais de distribuição [03,04]

Com a finalidade de distribuir o serviço prestado pelo tVoice, quanto ao recebimento e tratamento de múltiplos clientes de forma concorrente, procurando atendê-los independentemente da localidade de funcionamento do mesmo e já antecipando, futuramente, a perspectiva de atender diferentes tipos de sistemas de interfaces de áudio, neste trabalho optamos por utilizar o modelo arquitetural cliente-servidor para a prestação desses serviços.

A justificativa para esse fato dá-se pela ampla utilização do modelo no ambiente Web, ambiente em que se insere o tVoice. Também por, em alguns casos, ambos os modelos, de Pares e de Filtros, serem vistos como instâncias do modelo Cliente-Servidor [02,04].

2.1.1 Arquitetura Cliente-Servidor

A arquitetura Cliente-Servidor é caracterizada pela disposição dos componentes do sistema em locais distintos, associada à classificação dos mesmos em componentes clientes e componentes servidores.

Como citado anteriormente, é classificado como Cliente o componente que realiza solicitação de serviços, oferecidos por um servidor. É denominado de Servidor o componente responsável por receber uma solicitação e processá-la, enviando ao final do processamento uma resposta ao cliente solicitante.

Este modelo arquitetural pode ser encontrado tanto em ambientes distribuídos quanto em ambientes não distribuídos.

Em ambientes não distribuídos os componentes clientes e servidores encontram-se em uma única máquina ou estação, como observado na Figura 3.

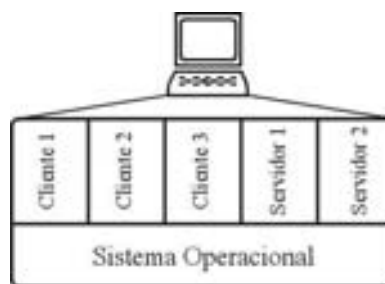



Figura 3 Modelo Cliente-Servidor em ambiente não distribuído [01]

Em ambientes distribuídos os componentes clientes e servidores encontram-se dispostos em máquinas ou estações diferentes, realizando a troca de informações através de mensagens que trafegam na rede. A Figura 4 apresenta uma visão simples de como se dá a disposição das estações na rede em um ambiente distribuído.

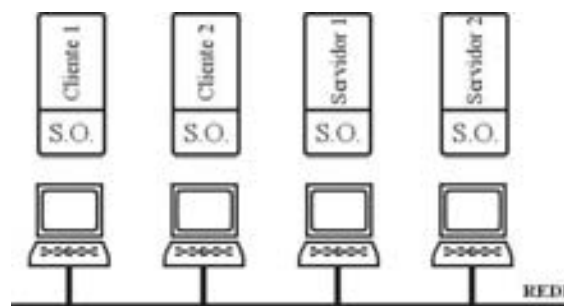



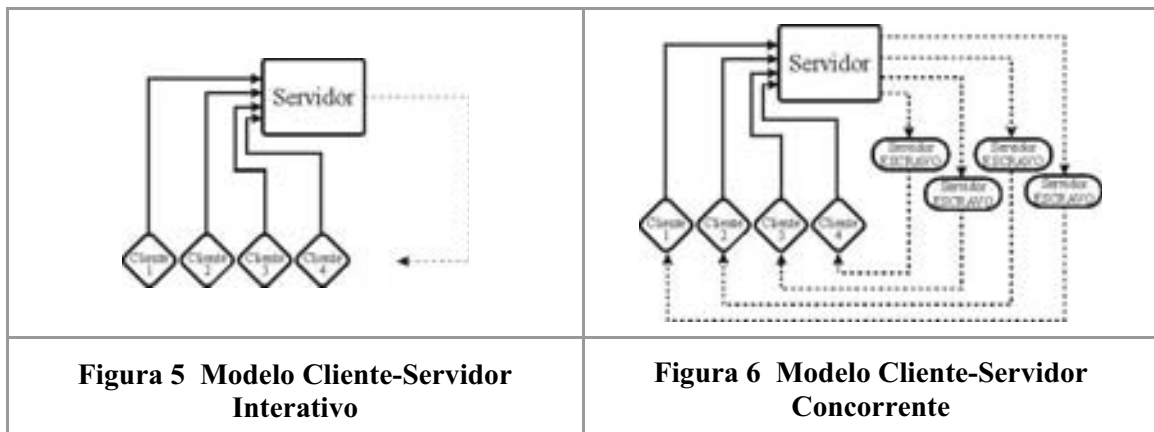
Figura 4 Modelo Cliente-Servidor em ambiente distribuído [01]

Tais conceitos não são obrigatórios para todos os sistemas, pois em um ambiente distribuído uma estação pode conter múltiplos clientes, múltiplos servidores ou combinações de ambos, permitindo que servidores possam ser clientes de outros servidores [01].

Servidores podem ser ainda classificados em função de outros fatores, tais como tipo de conexão e capacidade de atendimento às solicitações do cliente.

Quanto à capacidade de atendimento aos clientes, os servidores podem ser divididos em duas classes [01,02]:

- **INTERATIVOS:** Refere-se ao servidor capaz de atender a uma única solicitação por vez. Esse tipo de servidor é encontrado em ambientes onde se utiliza exclusão mútua quanto às políticas de acesso. Por exemplo, operação de escrita em bancos de dados. (Ver ilustração da Figura 5).
- **CONCORRENTES:** Refere-se ao servidor capaz de atender mais de uma solicitação por vez. Podem ser citados, como exemplo, os servidores Web quando do atendimento a diversas solicitações oriundas de diferentes *browsers* em espaços de tempo quase simultâneos. (Ver ilustração da Figura 6).

Figura 5 Modelo Cliente-Servidor Interativo	**Figura 6 Modelo Cliente-Servidor Concorrente**

No ambiente da Internet, em função do tipo de conexão que utilizam, os servidores podem ser classificados como:

- **SEM CONEXÃO:** Cliente não necessita abrir conexão com Servidor antes de realizar a transferência de dados. Na Internet, utilizam o protocolo de comunicação UDP (User Datagram Protocol).
- **COM CONEXÃO:** Cliente necessita abrir conexão com Servidor antes de transferir dados. Na Internet, utilizam o protocolo de comunicação TCP (Transmission Control Protocol).

Priorizando o desenvolvimento deste trabalho junto ao ambiente em que o mesmo se insere, a saber, a Web, optou-se pela utilização do protocolo de comunicação em uso na maioria das aplicações que rodam neste ambiente, o TCP/IP.

Outro fator que justificou sua utilização é o fato desse protocolo permitir, através de suas características, uma base concreta para a execução de diferentes tipos de aplicações [03,05,07].

2.1.2 Arquitetura da Internet TCP/IP

A arquitetura da Internet TCP/IP baseia-se em um serviço orientado à conexão, através da utilização do protocolo TCP (Transmission Control Protocol) que fornece um serviço confiável de transferência de dados, e em um serviço de rede não-orientado àconexão, fornecido pelo protocolo IP (Internet Protocol) [03,05,07].

A ênfase dada pela arquitetura TCP/IP é a interligação de diferentes tecnologias de redes, uma vez que a Internet é composta por diversos tipos de sistemas, máquinas e estruturas.

Essa arquitetura é formada basicamente por quatro camadas conceituais, apresentadas pela Figura 7 [05].

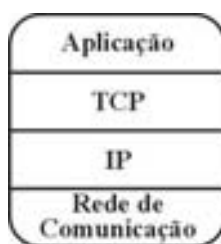



Figura 7 Camadas da arquitetura Internet TCP/IP

No nível ou camada de Aplicação, programas de aplicação são utilizados para acessar os serviços disponíveis nas camadas inferiores, abstraindo toda a complexidade envolvida nesta comunicação.

Segundo SOARES, LEMOS e COLCHER [05], algumas aplicações disponíveis na Internet TCP/IP são:

- SMTP (Simple Mail Transfer Protocol) - Serviço de mensagens.

- FTP (File Transfer Protocol) - Serviço de transferência de arquivos.
- TELNET - Serviço de terminal virtual.
- DNS (Domain Name System) - Serviço de mapeamento de nomes em endereços de rede.
- HTTP (Hypertext Transfer Protocol) - Serviço de transferência de documentos hipermídia.
- SSH (Security Shell) - Serviço de terminal virtual, semelhante ao Telnet, porém com maiores restrições de segurança.

Uma vez que o protocolo HTTP é mundialmente utilizado dentro do contexto da Internet, permitindo o fluxo de diversos tipos de conteúdo e de documentos entre diferentes tipos de aplicações e por ser o protocolo no qual a Web é construída, optou-se pela utilização do mesmo no processo de comunicação dos subsistemas. Tal fato justifica a apresentação mais detalhada do mesmo nas seções seguintes.

2.1.3 Protocolo de Comunicação HTTP

O HTTP (Hypertext Transfer Protocol) é um protocolo usado para transferência de informações na Internet. Os dados transmitidos através da utilização do mesmo podem ser de vários tipos, tais como texto, hipertexto, imagens, sons, dentre outros, possibilitando a transferência de uma gama imensa de informações [05,08,09].

SOARES, LEMOS e COLCHER [05] escrevem que esse protocolo permite o acesso a um amplo número de formatos de documentos, fato que o torna protocolo padrão utilizado pelos *browsers* na Web.

Seu funcionamento é bastante simples e muito eficiente. Utilizando solicitações feitas em codificação ASCII[2] e respostas do tipo MIME[3], possibilitam a circulação de diversos tipos de mídia pela rede.

Com a utilização desse protocolo, o servidor não necessita armazenar informações do cliente, aguardando apenas novas solicitações na porta 80, porta esta especificada como padrão para comunicação utilizando este protocolo [08,09].

---

[2] **ASCII** - American Standard Code for Information Interchange. Sistema de codificação norte-americano, mundialmente utilizado para representar computacionalmente símbolos de linguagem.
[3] **MIME** - Multipurpose Internet Mail Extensions. Permite a extensão dos formatos das mensagens trocadas na Internet, ampliando os tipos de conteúdo que nas mesmas trafegam.

Dentre as primitivas suportadas por esse protocolo em um servidor, podem ser destacadas [08]:

- **GET** - primitiva utilizada na solicitação de um documento.
- **HEAD** - primitiva utilizada na solicitação do cabeçalho de um documento.
- **POST** - primitiva utilizada na solicitação de postagem de dados no servidor.

Um exemplo de modelo de uma solicitação HTTP 1.0 é apresentado pelo quadro 1, a seguir.

Quadro 1 Modelo de solicitação HTTP 1.0

EXEMPLO DE GET	EXEMPLO DE RESPOSTA DO SERVIDOR
GET <caminho><documento> HTTP/1.0 Host: <endereço destino> Accept: <tipo de documento> Accept-Language: <linguagem> Accept-Encoding: <tipo de codificação> User-Agent: <browser> <linha em branco>	HTTP/1.0 <código de resposta> OK Date: <data recebimento solicitação> Server: <servidor> Last-Modified: <data de modificação do documento> ETag: <identificador> Accept-Ranges: <unidade> Content-Length: <tamanho> Content-Type: <tipo do conteúdo> <linha em branco> <código do documento>

Os objetos Web são endereçados através de *strings* denominadas URIs[4], utilizados na localização dos servidores. Um exemplo simples de solicitação HTTP, realizada através de um *browser* comum, pode ser expresso da seguinte forma:

[4] **URI** - Universal Resource Identifiers. Responsável pela identificação única de uma máquina servidora na Internet.

2.2 Compiladores

Segundo SETZER e MELO [11], um compilador é um programa que tem por finalidade traduzir e converter um código/programa escrito em uma linguagem Lf, chamada de fonte, em um código/programa escrito em outra linguagem Lo, chamada de objeto.

Tais linguagens são também associadas ao nível de programação em que trabalham, sendo assim classificadas ou denominadas de alto nível e baixo nível, respectivamente [11,12].

Observando um compilador através de uma macro-visão pode-se visualizar a seguinte estrutura, apresentada pela Figura 8 [10,11].

Figura 8 Macro-visão de um compilador

O processo de compilação, segundo AHO, SETHI e ULLMAN [10], seguindo o modelo de Análise e Síntese, pode ser dividido em duas partes. A Figura 9 apresenta as partes do modelo de Análise e Síntese e suas subseqüentes divisões, permitindo uma visão mais detalhada da forma geral de um compilador.

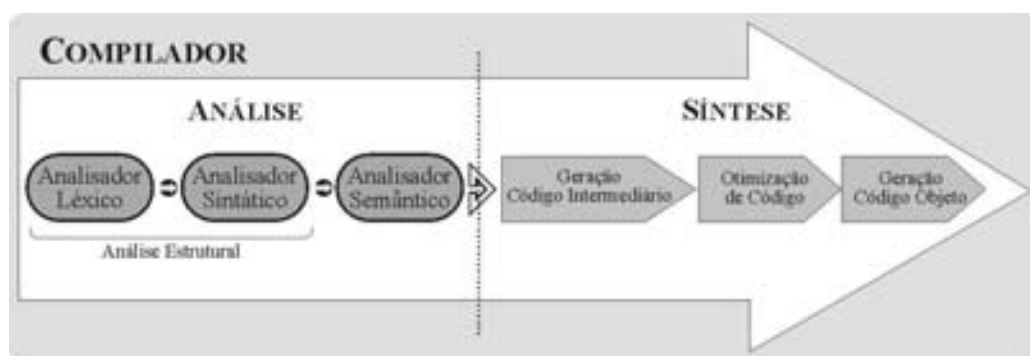



Figura 9 Divisão conceitual de um compilador

Do modelo conceitual apresentado, as fases de um compilador empregadas e implementadas no projeto de realização deste trabalho foram: análise léxica, análise sintática e análise semântica, tendo ao final, a geração de código direta, sem que

houvesse a necessidade de se prosseguir nas fases seguintes desse modelo conceitual de compiladores.

Tal fato se deu em função do trabalho desenvolvido, objetivar a conversão direta de uma linguagem fonte em uma outra linguagem alvo equivalente, baseando-se na estrutura da formatação do código fonte.

Partindo desse ponto e abstraindo as subdivisões encontradas dentro do ambiente de estudos de compiladores, oriundas das diferentes visões e teorias que envolvem esse assunto, as fases de análise léxica, sintática e semântica utilizadas podem assim ser apresentadas.

2.2.1 Fases de uma Compilação

Mesmo não sendo regra geral, anteriormente à construção de um compilador é necessário que se escreva uma estrutura gramatical que especifique a configuração da linguagem fonte, essa estrutura é chamada de gramática livre de contexto ou BNF (Backus-Naur Form). Essa gramática regulamenta como os componentes léxicos e sintáticos devem se comportar dentro da linguagem fonte [10,11,12,24].

As gramáticas livres de contexto podem ser escritas, de uma forma mais intuitiva para que sejam melhor visualizadas e entendidas, através da utilização da técnica chamada de regras de produção, onde podem ser observados os componentes léxicos e sintáticos dentro da linguagem.

Na prática, comumente associadas às regras de produção estão as expressões regulares que definem as construções da linguagem. Através destas podem ser descritas as produções cujos elementos incluem símbolos terminais (que fazem parte do código fonte) e símbolos não-terminais (que geram outras regras). Comumente as classes de símbolos não-terminais são definidas através de outras expressões regulares mais simples.

Depois de definida a gramática da linguagem fonte, a primeira fase ou passo de um compilador é a análise léxica, responsável por acessar seqüencialmente o código fonte, agrupando os símbolos de cada item léxico e determinando sua respectiva classe, inserindo-os, quando necessário, em uma estrutura de dados chamada de tabela de

símbolos. Isto é, converter um fluxo de caracteres de entrada em um fluxo de segmentos, denominados de *tokens*, servindo como entrada para fase seguinte [10,11,24].

Os espaços que separam os caracteres desse *tokens* são geralmente eliminados durante o processo realizado nessa fase, uma vez que somente os *tokens* importam para a fase seguinte, salvo quando se deseja conservar o "layout" do código fonte.

Um exemplo simples de análise léxica pode ser observado na Figura 10, abaixo [10].

ENTRADA	TOKENS
`montante := deposito + taxa_juros * 60`	Identificador **`montante`** Símbolo de atribuição **`:=`** Identificador **`deposito`** Operador de adição **`+`** Identificador **`taxa_juros`** Operador de multiplicação **`*`** Número **`60`**

Figura 10 Exemplo - Análise Léxica

É ainda função da análise léxica reportar os erros encontrados na varredura do código fonte, isto é, erros relativos a seqüências de caracteres que não correspondem a nenhum tipo de *token*.

A fase seguinte à análise léxica é, como mostrado na Figura 9, a análise sintática ou análise hierárquica, ou ainda, análise gramatical, responsável por verificar se o fluxo/seqüência de *tokens*, recebido da fase anterior, corresponde fielmente ao expresso na gramática que se objetiva. Caso isso não seja confirmado um erro sintático deve ser reportado pelo compilador [10,11,13].

É comum representar-se os componentes sintáticos ou gramaticais de um código fonte utilizando-se uma estrutura gráfica chamada de árvore de derivação. Para a entrada do exemplo apresentado pela Figura 10 temos a seguinte árvore, visualizada na Figura 11.

Figura 11 Árvore gramatical - exemplo Figura 10

Depois de realizadas as verificações léxicas e sintáticas, o compilador deve executar a fase denominada de análise semântica, que irá verificar se o significado das construções sintáticas expressa o sentido real do que deveria, reportando, quando houverem, os erros contidos no programa e capturando informações necessárias para possibilitar a realização da fase de Síntese.

A fase de Síntese, priorizando a simplificação, pode-se resumir à geração de código intermediário, uma vez que, seguindo a teoria da subdivisão da compilação, as fases seguintes a esta estão diretamente associadas ao hardware em que se irá trabalhar [10].

Partindo dessa perspectiva, pode-se apresentar a fase de geração de código como sendo a fase final de um compilador que, de posse das informações coletadas durante as fases anteriores, gerará um código objeto/destino.

2.2.2 Técnicas Utilizadas em Desenvolvimento

Para a construção de compiladores são utilizadas algumas técnicas que podem ser classificadas pelo número de passagens pelo código e pelo tipo de análise utilizada na passagem.

Quanto ao número de passagens, pode-se utilizar uma, duas ou mais, dependendo de como se processará o código ou programa fonte.

Nos compiladores mais simples, caso este do utilizado neste trabalho, de uma só passagem, também conhecido como tradução dirigida pela sintaxe, todas as fases

da compilação são realizadas no momento da passagem, tendo ao final da mesma o código ou programa objeto.

Já nos compiladores de duas ou mais passagens, a cada passagem é gerada uma representação intermediária entre as mesmas, que vão servindo de entrada para as passagens ou fases seguintes.

Tanto nos compiladores de uma ou mais passagens pode-se utilizar dois tipos de análises, associados à análise sintática. São elas:

- **TOP-DOWN** – é uma análise onde se procura, a partir do símbolo de partida da gramática, chegar à cadeia que está sendo analisada, progredindo nas regras de produção. Utilizando-se uma estrutura gráfica de árvore para a representação da análise, diz-se que esta é feita da raiz para as folhas.
- **BOTTOM-UP** – é uma análise onde se procura, a partir da cadeia que está sendo analisada, chegar ao símbolo inicial da gramática, regredindo nas regras de produção. Utilizando-se uma estrutura gráfica de árvore para a representação da análise, diz-se que esta é feita das folhas para a raiz.

Além da possibilidade de se construir compiladores a partir do "zero", ou seja, desenvolver todas as suas fases, implementando cada uma delas desde a leitura dos caracteres, passando pelo reconhecimento de *tokens*, analisando sua sintaxe e significação semântica e finalmente gerando um código objeto, é ainda possível utilizar-se ferramentas que facilitam e tornam o trabalho de desenvolvimento de compiladores mais prático e rápido.

Através do uso de geradores de *parser*, como são mais conhecidos, ou compilador de compiladores, o trabalho de desenvolvimento é simplificado, embora não seja possível sua utilização no tratamento de qualquer tipo de gramática, empregando ainda certos tipos de restrições às gramáticas reconhecidas, dependendo da classe a que pertencem.

Existem ainda casos em que tais ferramentas não podem ser utilizadas, em função das restrições impostas pelas gramáticas, criando-se a necessidade de se

desenvolver cada uma das fases da compilação separadamente, comumente chamado de desenvolvimento "à mão".

No desenvolvimento de alguns dos módulos desse trabalho, foi utilizada a tecnologia de geradores de *parser* na construção dos tradutores empregados na implementação do mesmo, justificando uma melhor explanação sobre o assunto nas seções seguintes.

2.2.3 Ferramentas de Geração de Parser

Existem várias ferramentas utilizadas para auxiliar no desenvolvimento de compiladores, permitindo rapidez, simplicidade e bons resultados. Dentre as mais conhecidas e utilizadas podem ser destacadas:

- **LEX/YACC** - Desenvolvidas separadamente, porém utilizadas em conjunto para funcionarem como uma ferramenta para a construção de compiladores. São respectivamente geradores de analisadores léxicos (Lex - lexical) e sintáticos (Yacc - Yet Another Compiler Compiler). Permitindo inserção de código em seus escopos, produzem ações, incluindo outros tipos de análises e sínteses [10]. Atualmente podem ser encontradas versões para as mais variadas linguagens de programação, tais como: Pascal, C++ e Java. As classes de linguagens reconhecidas são do tipo LALR(1)[5].
- **JLEX/JAVACUP** - Desenvolvidas para trabalhar com a linguagem de programação Java, são também geradores de analisadores léxicos (Jlex) e sintáticos (JavaCup) que, assim como os outros, permitem a inserção de código para realização das fases seguintes as de análises [30]. As classes de linguagens reconhecidas são do tipo LR(1)[6].
- **JAVACC** (Java Compiler Compiler) - Ferramenta que engloba a geração de analisadores léxicos e sintáticos em um só ambiente.

---

[5] **LALR(k)** - LookAhead LR. Classe de linguagens derivadas da classe LR, porém com custo de construção menos elevado, onde k é a quantidade de símbolos de entrada que são utilizados nas tomadas de decisão na análise sintática.

[6] **LR(k)** - Sigla significando a contração de varredura de entrada da esquerda para a direita (do inglês Left-to-right) e construção de derivação mais à direita (do inglês Right most derivation). O "k" representa o número de símbolos de entrada usados na tomada de decisão na análise sintático.

Também construída para trabalhar sobre a plataforma Java de desenvolvimento, esta ferramenta permite a inserção de código em seu escopo, gerando, ao final, um conjunto de instruções Java capaz de ser importado e utilizado por instâncias de outras classes de objetos [14,15]. Por padrão, reconhece classes gramaticais do tipo LL(1)[7], podendo em alguns pontos assumir características de LL(k) através de opções internas.

Em função de o JavaCC, versão 2.0, ter sido a ferramenta adotada para desenvolvimento de módulos do tVoice, a seção seguinte apresenta esta ferramenta em maiores detalhes.

2.2.3.1 A Ferramenta JavaCC

Nos arquivos de gramática do JavaCC, por padrão gerador de gramáticas LL(1), os *tokens* seguem as mesmas convenções da linguagem de programação Java, logo, também os identificadores de *strings* e outros componentes usados na construção das gramáticas se tornam os mesmo utilizados na linguagem. Os comentários no JavaCC possuem a mesma sintaxe utilizada nos arquivos Java e os arquivos de gramática são também pré-processados para codificação Unicode [15].

JavaCC, utilizando uma linguagem própria para a representação de alguns de seus componentes, possui palavras reservadas que são utilizadas na construção de certas definições. Tais palavras são: EOF, IGNORE_CASE, JAVACODE, LOOKAHEAD, MORE, options, PARSER_BEGIN, PARSER_END, SKIP, SPECIAL_TOKEN, TOKEN e TOKEN_MGR_DECLS [14,15].

A estrutura de um arquivo JavaCC possui a seguinte formatação e distribuição:

```
Options {
    [OPÇÕES]
    ...
}
```

[7] **LL(k)** - Sigla significando a contração de varredura de entrada da esquerda para a direita (do inglês Left-to-right) e construção de derivação mais à esquerda (do inglês Left linear). O “k” representa o número de símbolos de entrada usados na tomada de decisão na análise sintática.

```
PARSER_BEGIN([NOME_PARSER])
...
    public class [NOME_PARSER] {
    ...
    }
PARSER_END([NOME_PARSER])

... //Produções da gramática (análise léxica)
... /* funções de representação das produções
      (sintático + semântico + ações)*/
```

As opções, utilizadas em um documento JavaCC, podem ser especificadas no início do arquivo ou nas linhas de código das produções. Se uma ocorrência do segundo caso é encontrada, ela tem precedência sobre uma ocorrência do primeiro, ou seja, têm prioridade as especificações feitas nas produções. Tais ocorrências, sem exceções, devem ser escritas em letras maiúsculas.

Das opções possíveis, podemos citar algumas opções utilizadas na realização do trabalho [14,15]:

- **LOOKAHEAD:** Especifica o número de *tokens* que devem ser lidos antes da tomada de decisão durante o *parsing*. Seu valor padrão é 1.
- **STATIC:** Opção booleana de valor padrão *true*, que indica que todos os métodos e variáveis da classe do *parser* são estáticos tanto no *parser* quanto no gerenciador de *tokens*.
- **DEBUG_PARSER:** Opção booleana de valor padrão *false*, que desabilita a geração do rastreamento das ações do *parser*.
- **DEBUG_TOKEN_MANAGER:** Opção que permite habilitar ou desabilitar a geração de informações de depuração do gerador de *tokens*. Seu valor padrão é *false*.
- **IGNORE_CASE:** De valor padrão *false*, faz com que o gerenciador de *tokens* não faça distinção entre letras maiúsculas e minúsculas.
- **SANITY_CHECK:** Possui valor padrão *true*, que habilita verificações sintáticas e semânticas, tais como descoberta de recursão àesquerda e ambigüidade, durante a geração do *parser*.

- **OUTPUT_DIRECTORY:** Opção do tipo String que controla o local onde serão gerados os arquivos de saída.

Como visto e exemplificado anteriormente, as produções seguem as declarações de início e fim da classe base para a geração de um *parser*. As produções de expressões regulares, escritas em JavaCC, podem ser de um dos seguintes tipos:

- **SKIP** – Neste tipo de produção de expressões regulares, são ignoradas pelo gerenciador de *tokens* todas as expressões nele contidas.
- **TOKEN** – As expressões regulares encontradas neste tipo de produção descrevem *tokens* que possuem significado na gramática, durante o *parsing*.
- **SPECIAL_TOKEN** – Semelhantemente aos TOKENS, as expressões regulares presentes neste tipo de produção de expressões descrevem *tokens*, porém sem significado durante o *parsing*.
- **MORE** – Útil quando se quer construir um *token* para ser passado para o *parser* gradualmente, sendo armazenados em um buffer até o próximo casamento de TOKEN ou ESPECIAL_TOKEN.

A seguir podemos visualizar um exemplo simples de declarações de expressões regulares.

```
...
SKIP:{
      < (" " | "\t" | "\n")+ >
  |   < ("abc") >
}

TOKEN:{
      < NUMERO: ( <DIGITO> )+ >
  |   < #DIGITO: ["0" - "9"] >
  |   < PALAVRA: (["a" - "z", "A" - "Z"])* >
}

void escrita():  //primeira função a ser chamada
                 //quando da instanciação do parser
{ token t; }     //atributos em Java
{
      { /* códigos ou ações semânticas opcionais */ }
  ( ( t=<PALAVRA> | <NUMERO> ) esp() )*
```

```
        { /* códigos ou ações semânticas opcionais*/ }
}

void esp():
{}
{       { /* códigos ou ações semânticas opcionais */ }
    ( " " | "\t")*
        { /* códigos ou ações semânticas opcionais */ }
}
```

O símbolo "#", antes da declaração do nome de uma produção, indica que este *token* só é válido dentro do escopo de definição, servindo apenas para o auxílio na definição de outro *token* [14,15].

O gerenciador de *tokens* pode ainda redirecionar o reconhecimento para uma outra área de definições, determinando-a através da utilização das propriedades chamadas de expansões. Para tal, basta que a declaração da produção seja seguida pela cadeia: "**:**" **[nome da expansão]**. A nova área de definições deve ser inicialmente identificada pela mesma cadeia associada e prefixada pelo símbolo "<" e sufixada pelo símbolo ">".

Junto às escolhas das produções limitadas por parêntesis podem ser encontrados os símbolos "+", "*", "?" indicando respectivamente que [15]:

- É válido o conjunto de uma ou mais repetições.
- É válido o conjunto de nenhuma ou várias repetições.
- É válido o conjunto de ou uma ou nenhuma ocorrência.

Existe ainda uma classe chamada "Token" que é instanciada por objetos do tipo "token", criados pelo gerenciador de *tokens* depois de um casamento de caracteres encontrado no fluxo de entrada. Esses objetos possuem, assim como em Java, métodos e atributos que podem ser acessados pelas ações gramaticais escritas dentro do corpo das produções. Os métodos são: getToken e getNextToken. Os atributos comuns a todos os objetos token são:

- **int** kind, beginLine, beginColumm, endLine, endColumm
- **String** image
- **Token** next, specialToken,

Subseqüentemente as fases de escrita e edição, do conteúdo formador do compilador que se deseja produzir com o JavaCC, os arquivos produzidos, possuidores de extensão ".jj", após serem processados e compilados, são transformados em classes Java que podem, assim, ser utilizadas e instanciadas por outras classes, possibilitando o uso do compilador gerado em outras localidades, dentro das mesmas.

3. Trabalhos Correlatos

Neste capítulo são apresentados esforços de pesquisa, alguns no âmbito da Acessibilidade, onde são utilizadas linguagens/documentos fonte no processo de extração/transformação de informação. Nesse contexto insere-se o trabalho desenvolvido e apresentado nessa dissertação.

3.1 Projeto AHA

O projeto AHA (Audio HTML Access) [21] realizou um estudo detalhado sobre mecanismos de apresentação de conteúdo HTML usando a mídia áudio. Para realização deste estudo, foram feitos diversos testes com diferentes tipos de usuários, utilizando vários recursos para a apresentação da informação, tais como a utilização de múltiplas vozes para as estruturas do documento, padrões de voz específicos para cada elemento definido no HTML, além da utilização de outros recursos sonoros.

O trabalho promove, ainda, uma discussão comparativa entre duas categorias de interfaces de áudio, as baseadas em sons diversificados e sons característicos, bem como a utilização de ambas em um sistema de auxilio a deficientes visuais.

A primeira delas, baseia-se na utilização de sons que apenas indiquem ao usuário o acontecimento de um evento, não importando se o som tenha um significado cognitivo para o usuário.

Já na segunda categoria de interfaces, defende-se a utilização de sons que façam algum sentido na mente do usuário, quando de sua utilização do mesmo, fazendo com que haja uma associação mental com sons encontrados pelos usuários em seu cotidiano. Por exemplo, o som do amassar de uma folha de papel quando se manda algo para a lixeira, ou o som de um folhear de páginas ao se passar para outra página de um texto.

Este projeto baseia-se no princípio de que arquivos HTML contenham explicitamente o conteúdo textual e estrutural em um só documento, e que, estes dois tipos de conteúdo juntos são essenciais para a compreensão e apresentação do documento.

Esse estudo, realizado no projeto, fornece uma visão essencial para o desenvolvimento de processadores de arquivos HTML utilizados no auxílio a interfaces de áudio, fornecendo uma base para o preenchimento de alguns dos requisitos de interação com os deficientes visuais.

A transformação ou tradução do conteúdo extraído baseia-se na estrutura sintática e semântica da linguagem HTML, realizando o mapeamento direto da linguagem e associando sons distintos a cada elemento encontrado no documento. Exemplos de elementos para os quais existem sons característicos são as imagens, listas, tabelas, formulários, links e diferentes estilos de texto.

A proposta do AHA para a interação com o deficiente visual é baseada na utilização do teclado, do mouse e da síntese de voz. Atualmente, o projeto possui um protótipo desenvolvido especificamente para a realização de testes, demonstrando a real intenção do projeto de funcionar como um *framework* de auxílio àcriação de interfaces e sistemas de processamento de documentos HTML.

Através da Figura 11, podemos visualizar a disposição do sistema AHA, tendo a compreensão de seu posicionamento em relação ao usuário e à Web.

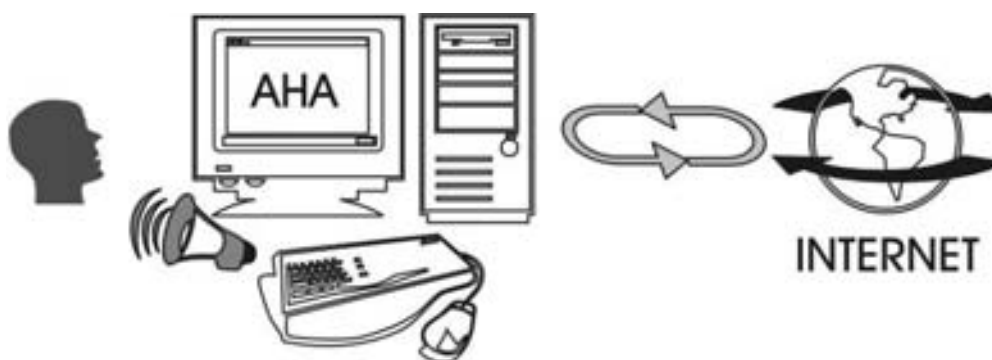



Figura 12 Disposição do sistema AHA

Assim esse projeto tounou-se base para o processo de definição das marcações de áudio a serem utilizadas no projeto VoiceProxy.

3.2 WAB

O sistema WAB [22] (Web Access for Blind users) foi desenvolvido, segundo seus idealizadores, para facilitar a navegação de deficientes visuais na Web, se interpolando entre o usuário e a Internet, assemelhando-se ao funcionamento de um servidor proxy.

No WAB, para que o usuário alvo navegue na rede, o sistema efetua uma reorganização no documento solicitado, durante a extração de informação. Essa ação é necessária, em virtude da estruturação e disposição das informações contidas em um documento HTML estarem voltadas, em quase toda a sua totalidade, à apresentação visual, dificultando, assim, o reconhecimento e extração através de sistemas de apoio aos deficientes visuais.

Durante o processamento das páginas HTML e extração das informações, a reorganização estrutural processa-se através da disposição dessas informações em uma estrutura própria, separando-as em: títulos, subtítulos, links, formulários, elementos de formulários, dentre outros elementos, obedecendo a uma ordem de descrição textual hierárquica.

São dois os passos do processo envolvido na manipulação dos documentos. O primeiro recupera cada elemento HTML transformando-o e ordenando-o de acordo com uma classificação hierárquica. O segundo passo insere marcações adicionais de controle dentro do novo documento gerado, permitindo o deslocamento do texto, na área de visualização do *browser*, para um novo local desejado dentro do documento solicitado.

Para que o usuário tenha acesso ao novo documento gerado, contendo as informações estruturadas e possibilitando sua navegação na Web, em sua máquina deve ter instalado um sistema chamado de "leitor de telas" — software que transforma as informações visuais contidas na tela do computador em informações não visuais, através da utilização da síntese de voz ou de impressoras braile.

A interação do usuário com o sistema se dá através da utilização do teclado e/ou mouse, em virtude do sistema não utilizar o reconhecimento de voz como forma alternativa de interação.

A disposição do sistema WAB, em relação à Web e ao usuário alvo, pode ser visualizado na Figura 12, a seguir.

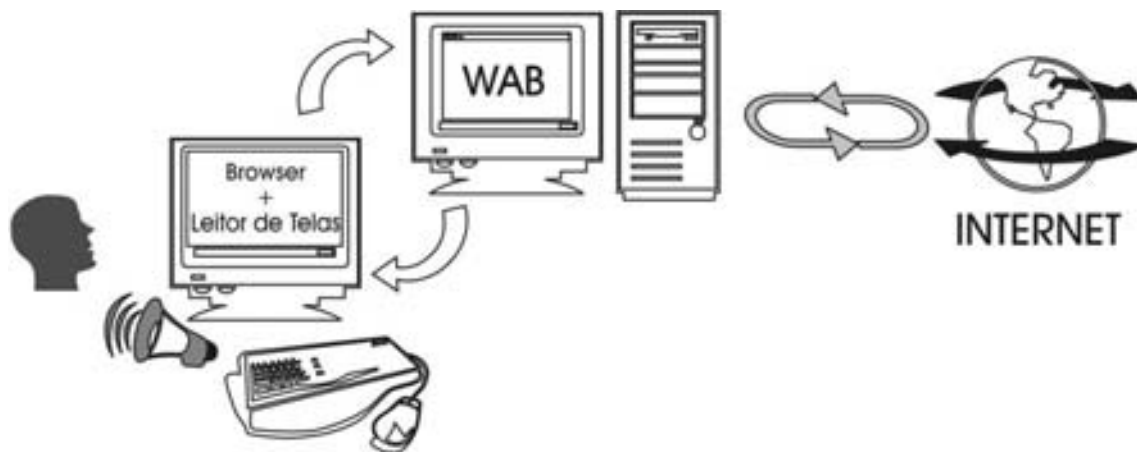



Figura 12 Disposição do sistema WAB

3.3 Audio XML

O Audio XML [23] foi um projeto desenvolvido na universidade da Califórnia, EUA, com propósito de promover a Acessibilidade, fornecendo condições de acesso aos serviços de informação, documentação e comunicação, por parte de portador de necessidades especiais, e a Computação Ubíqua, permitindo que a informação seja acessada por meio de direntes tipos de dispositivos, tentando assim disponibilizar a todos o acesso à Web.

É fato que tanto na Web quanto na maioria dos sistemas utilizados nos computadores pessoais, são empregados modelos de apresentação puramente visuais, o que acarreta uma série de problemas para aqueles que possuem algum tipo de necessidade especial, tais como os portadores de deficiência visual, que mesmo através do uso dos chamados leitores de tela, encontram dificuldades em obter e manipular

informações, uma vez que o áudio é um tipo de mídia serial, ou seja, de uma única dimensão.

Tendo como principal objetivo utilizar a estrutura, bem formada e extensível, da linguagem de marcação XML para, juntamente com uma interface baseada na interação através de áudio, proporcionar ao usuário portador de necessidades especiais a possibilidade de acesso, navegação, produção e modificação de informação, na Internet, o trabalho propôs e desenvolveu um conjunto de marcações próprias, destinadas à coordenação e gerenciamento da “leitura” e manipulação das informações, separando o conteúdo do documento da forma como deve o mesmo ser apresentado.

Nesse projeto, os usuários alvo do sistema desenvolvido são os portadores de deficiência visual e usuários que, em função das restrições impostas pela circunstância, não possam fazer uso do computador de maneira convencional, ou seja, através do monitor, teclado e mouse.

Utilizando o software ViaVoice da IBM, associado ao seu sistema, o Audio XML pode promover a interação com o usuário através da síntese e do reconhecimento de voz, permitindo assim o acesso às informações e a manipulação de dados.

Na Figura 13, encontra-se disposta a situação do sistema Audio XML em relação ao usuário alvo e à Web.

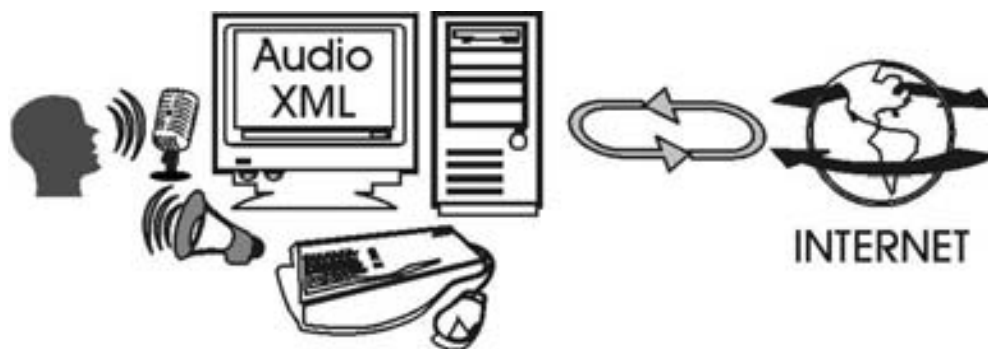



Figura 13 Disposição do sistema Audio XML

4. O Sistema VoiceProxy

O projeto VoiceProxy tem como principal meta, desde sua primeira versão, desenvolvida em 2001, promover a Acessibilidade dentro do ambiente Web [16]. Tentando minimizar as barreiras encontradas pelos deficientes visuais no acesso às informações disponibilizadas através de documentos HTML, o sistema utiliza a síntese e o reconhecimento de voz como método de acesso para os portadores de necessidades especiais, mais especificamente os deficientes visuais.

Em sua primeira versão, resumidamente descrevendo sua funcionalidade, o sistema funcionava instalado na máquina do usuário, semelhantemente a um *browser*, que depois de configurado e posto em operação, aguardava por um comando do usuário solicitando-lhe acesso a uma determinada página Web, seja pela fala (reconhecimento de voz), através de um microfone, ou teclado. Em seguida, o VoiceProxy buscava a página solicitada, realizava a extração do conteúdo textual da mesma e o lia (síntese de voz) para o usuário [16].

A Figura 14 nos apresenta a disposição da primeira versão do sistema VoiceProxy em relação ao usuário e à Web.

Figura 14 Disposição VoiceProxy 1.0

Mais recentemente, esse projeto teve outros desafios a superar, o que levou ao agenciamento da construção de uma nova versão do sistema, buscando, dentro de seus objetivos, ampliar suas características para melhor atender aos usuários,

permitindo que além do HTML, outros tipos de documentos possam ser acessados e utilizados, além de pretender que o sistema possa ser acessado remotamente.

Para a nova versão que atualmente ainda se encontra em desenvolvimento, pretende-se uma mudança física na disposição do VoiceProxy, sem que haja qualquer alteração na disposição lógica, permitindo que o usuário não perceba a diferença de localização do sistema, uma vez que a idéia é que, como dito, o mesmo possa ser utilizado remotamente como uma página Web.

A Figura 15 apresenta a nova disposição física do VoiceProxy em relação ao usuário e a Web, tornando o sistema mais acessível e independente de alguns fatores, tais como sistema operacional do usuário.

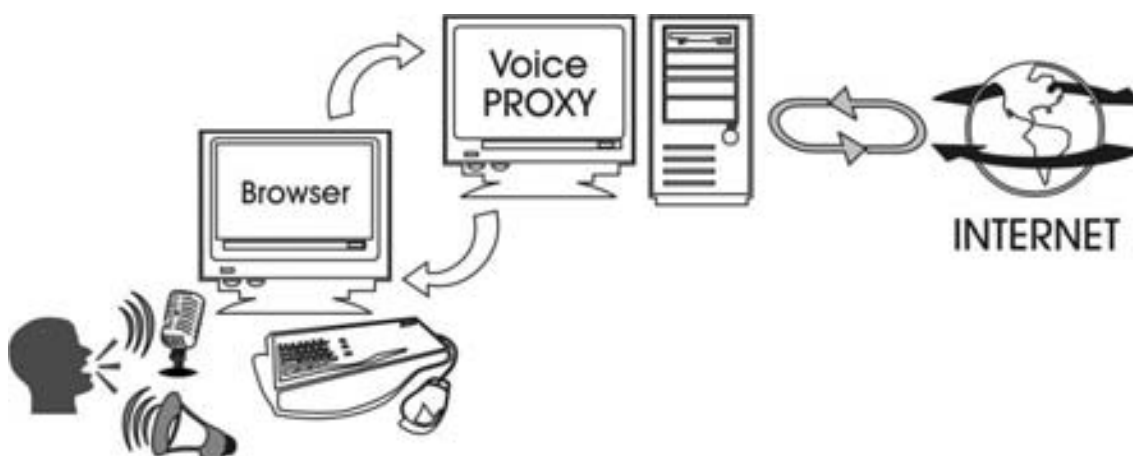



Figura 15 Disposição VoiceProxy 2.0

Nesta nova etapa o sistema VoiceProxy ainda está subdividido em dois subsistemas distintos, que trabalham em conjunto para atingir suas metas. São eles:

- **IVOICE (SUBSISTEMA DE INTERFACE)** – responsável pela interação com o usuário através da síntese e reconhecimento de voz ou teclado, e também pela navegação do usuário na Internet. Funcionará através de um *browser* visual comum e será acessado de forma semelhante a uma página Web, tornando-o mais acessível e independente de plataforma.
- **TVOICE (SUBSISTEMA DE TRADUÇÃO)** - responsável pela comunicação com o servidor ou repositório Web em procurando recuperar os documentos solicitados, juntamente com a extração e

processamento de informações contidas nesses, sendo atualmente capaz de processar os seguintes formatos: HTML, XML e PDF. Atualmente funciona como um servidor passivo, aguardando uma solicitação de processamento, retornando ao final um documento chamado de script de áudio (*tags* de áudio mais conteúdo textual extraído).

Na Figura 16 podemos visualizar, de forma geral, a subdivisão estrutural do sistema VoiceProxy.

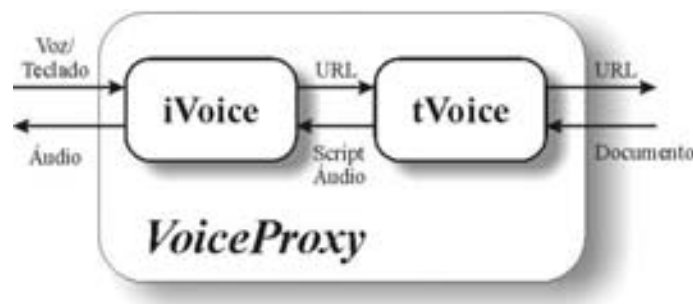



Figura 16 Divisão do VoiceProxy em Subsistemas

O funcionamento do sistema VoiceProxy pode ser também descrito através do seguinte cenário, apresentado pelo Quadro 2.

Quadro 2 Cenário funcionamento VoiceProxy

Cena 1. Usuário tem acesso ao sistema *O usuário alvo do sistema (deficiente visual), com o seu computador já configurado para o acesso à Internet e tendo no seu browser, como página inicial de acesso, o endereço do servidor do sistema VoiceProxy, aguarda a inicialização do sistema em seu browser.* *Depois de carregada a página de acesso no browser do usuário, o sistema o avisa que está pronto para iniciar a interação, sintetizando em áudio, através do sistema de som do computador, a frase: "VoiceProxy inicializado!".*
Cena 2. Usuário solicita ajuda. *O usuário iniciante, sem conhecer toda a funcionalidade do sistema, pronuncia no microfone instalado em sua máquina a palavra "ajuda", solicitando que o sistema o informe a respeito de como utilizá-lo.* *Processado o comando, o sistema inicia a "leitura" do conteúdo do manual de uso rápido do sistema, para o usuário.*

Ao escutar a passagem desejada de como se realizar a tarefa almejada, o usuário pede ao sistema que pare a leitura do manual, pronunciando a frase: "parar leitura".

Cena 3. Usuário solicita documento na Web.

O usuário aciona o sistema pronunciando, no microfone instalado em seu computador, o comando: "navegar", disponível na biblioteca do sistema, equivalente a uma solicitação de recuperação de documento na Web.

Em seguida o usuário ouve a seguinte frase: "Favor informar o endereço eletrônico".

Assim sendo, o usuário pronuncia o endereço eletrônico em seu microfone, letra a letra, ouvindo a confirmação de cada uma delas. Ao terminar de ditar o endereço desejado, o usuário solicita a confirmação do mesmo através do comando: "confirmar", que dispara o evento de leitura de todo o link.

Recebida a solicitação, o sistema percorre a Web em tentando recuperar o documento solicitado.

Encontrado o documento, o VoiceProxy processa o mesmo e dele extrai o conteúdo informativo e o "lê" para o usuário através do sistema de som do computador.

Apesar do usuário alvo do VoiceProxy ser o portador de deficiência visual isso não impede que o sistema seja utilizado para atender outros tipos de usuários, como por exemplo pessoas incapacitadas de utilizar teclado e mouse, usuários de terminais de acesso que não dispõem destes dispositivos, motoristas, etc.

A proposta do VoiceProxy é tentar fazer com que o maior número de usuários especiais possam utilizar a Internet, sejam eles possuidores de maior ou menor capacitação para o uso da máquina, e assim, fazer com que eles possam ter na Internet uma fonte de informação crescente e inesgotável.

Assim como na primeira versão, o acesso dos portadores de necessidades especiais aos documentos, utilizados na extração de informação, ainda sofre restrições diretamente relacionadas aos tipos de documentos utilizados no processamento. Tais restrições e limitações, por estarem diretamente ligadas ao subsistema de tradução, parte integrante do sistema VoiceProxy, serão melhor apresentadas no capítulo seguinte.

5. *O tVoice - Subsistema de Tradução*

Nesta nova etapa do projeto VoiceProxy, o subsistema de tradução tem seu funcionamento distribuído, comunicando-se tanto com o subsistema de interface quanto com o repositório Web através do protocolo HTTP (HyperText Transfer Protocol) versão 1.0.

O protocolo utilizado possibilita que o tVoice trabalhe no ambiente Web, não só com o sistema VoiceProxy, mas permite a possibilidade futura de que o subsistema desenvolvido trabalhe com quaisquer sistemas de interface que façam uso de sintetizadores capazes de “ler” os documentos gerados. Para isso, teve-se como princípio inicial à utilização, na geração dos *scripts* de áudio, de *tags* de áudio propostas e padronizadas por comitês internacionais, dos quais faz parte a IBM [29], companhia desenvolvente do sintetizador utilizado no subsistema de interface do VoiceProxy.

Diante da idéia proposta para o novo subsistema de tradução do VoiceProxy, denominado de tVoice, iniciou-se uma nova fase de desenvolvimento do projeto, culminando com a construção de um subsistema de apoio ao subsistema de interface, cuja contribuição final foi a realização deste trabalho.

Na Figura 17 é apresentada a arquitetura base do tVoice, possibilitando uma macro-visão do seu funcionamento básico e do conteúdo das mensagens trocadas entre seus módulos.

Figura 17 Arquitetura do tVoice

O tVoice, ainda nesta nova versão, possui as seguintes funcionalidades básicas dentro do sistema VoiceProxy:

- Aguardar solicitações.
- Recuperar na Web os documentos solicitados.
- Processar e extrair informações textuais dos documentos.
- Gerar e enviar o Script de áudio ao solicitante.

Acrescido a essas funcionalidades, o tVoice baseia-se no modelo arquitetural Cliente-Servidor quanto à prestação de serviços na rede, utilizando ainda parte do modelo de Filtros em sua estrutura interna, funcionando no modo síncrono de transferência de dados.

O subsistema ainda utiliza o modo concorrente de atendimento, permitindo que múltiplas solicitações sejam atendidas separadamente, mas em simultaneidade.

Para possibilitar o processo de comunicação na rede, o protocolo HTTP 1.0 foi escolhido em função de sua simplicidade, alto desempenho na manipulação de diversos tipos de documentos e por ser o protocolo padrão da Web.

Dos métodos de comunicação encontrados no protocolo em questão, o tVoice implementa apenas o método GET, uma vez que, ainda nesta versão, o sistema processa somente solicitações de recuperação de documentos, sem realizar postagem de dados nos servidores.

O funcionamento geral do subsistema, pode ser descrito em passos, como apresentado pelo Quadro 3, a seguir.

Quadro 3 Passos do funcionamento tVoice

PASSOS	REALIZAÇÃO
1	Depois de iniciado, o subsistema aguarda por solicitações via HTTP, provenientes do subsistema iVoice.
2	O módulo de gerência se encontra pronto para disparar um servidor-escravo, de atendimento, para cada nova solicitação recebida.
3	Depois de aceita e encaminhada a um servidor-escravo, a solicitação é processada e uma conexão HTTP é estabelecida com o servidor Web no qual se encontra o documento solicitado.
4	Se encontrado na Web o documento solicitado, uma identificação do formato do mesmo é realizada, juntamente com a solicitação inicialmente recebida, a fim de prepará-lo para o processo de extração de informações textuais.
5	Identificado o formato do documento recebido, a tradução é iniciada permitindo que seja realizado um processamento no documento e dele seja extraída a maior quantidade de informações textuais possível, sendo gerado em seguida um documento chamado de ***script de áudio***.[8]
6	Executados todos os passos anteriores, o script de áudio é enviado ao solicitante utilizando-se novamente o protocolo HTTP 1.0.

[8] *Documento formado pela associação de tags/marcações de áudio com as informações textuais extraídas. Tais marcações guiam a síntese de voz do documento e são utilizadas pelo subsistema de interface na interação.*

O tVoice é composto por quatro módulos básicos, como visto na Figura 17, dentro de sua arquitetura básica.

O primeiro deles é o Módulo Gerente, ou de gerenciamento, que é responsável pelo aguardo e primeiro processamento no recebimento das solicitações. Funcionando no modo passivo, dispara novos servidores-escravo para cada solicitação recebida do subsistema de interface.

O Módulo Servidor, após ser iniciado, se responsabiliza completamente pela recuperação do documento solicitado e entrega do documento gerado. Este módulo utiliza-se do Módulo de Conexão para preparar o envio tanto da requisição GET quanto do *script de áudio* gerado, respectivamente enviados a um servidor/repositório Web e para o subsistema de interface.

O Módulo de Identificação/Tradução é responsável pelo processo de extração da maior quantidade de informação textual possível dos documentos recebidos. Esse módulo também efetua uma verificação junto à solicitação inicialmente recebida, para poder, conseqüentemente, preparar todo o ambiente necessário ao processamento de tradução do respectivo tipo de documento a ser tratado.

Nas seções que seguem foram utilizadas notações UML (*Unified Modelling Language*) com o propósito de modelar e documentar o projeto de desenvolvimento e atualização do tVoice, permitindo assim apresentar uma visão mais técnica e precisa das fases do projeto.

5.1 Análise de Requisitos

Nesta fase do projeto, partindo da construção da arquitetura e de alguns cenários idealizados, foram utilizados para descrever o sistema os seguintes diagramas UML: casos de uso, atividades, seqüência e classes.

A meta deste estágio foi determinar os resultados que o sistema deverá produzir, tendo o foco voltado para as necessidades das realizações e não na solução técnica que deve ser adotada. Isto é, os requisitos estabeleceram o que o sistema deverá fazer e não como fazer [18].

5.1.1 Diagramas de Casos de Uso

Em um projeto, os diagramas de casos de uso são utilizados para captar do sistema o comportamento pretendido, sem que seja necessário especificar como esse comportamento é implementado. Os casos de uso permitem que os designers possam chegar a uma mesma compreensão do funcionamento do sistema junto com os usuários finais e o especialista do domínio do problema. Além disso, tais diagramas auxiliam na validação da arquitetura [17].

Tentando observar o comportamento geral do sistema VoiceProxy, em relação aos extremos nos quais está interpolado, foi idealizado, em conjunto com os idealizadores do sistema, o seguinte diagrama de casos de uso, apresentado pela Figura 18.

Figura 18 Casos de Uso - Comportamento VoiceProxy

Conseguinte a esse diagrama, partindo da divisão lógica do VoiceProxy em subsistemas, como apresentado no Capítulo 4 desta dissertação, tendo ainda a pretensão de direcionar o foco para a realização do trabalho em tese; objetivamos representar o processo de inicialização e de finalização do funcionamento do subsistema tVoice, relacionando-o com o único ator (usuário) do conjunto, a saber, o administrador do servidor em que o subsistema residirá. Com esse fim idealizou-se o seguinte diagrama, apresentado pela Figura 19.

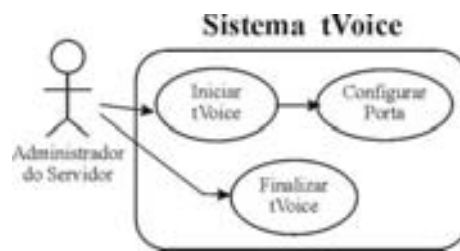



Figura 19 Casos de Uso - Inicialização e finalização do sistema tVoice

Subseqüentemente, foi construído o diagrama de casos de uso referente ao funcionamento geral do subsistema de tradução, denominado anteriormente de tVoice, podendo o mesmo ser visualizado na Figura 20.

Figura 20 Casos de Uso - Comportamento geral tVoice

Depois de identificadas as funções gerais do novo subsistema de tradução e de posse da nova rede de proposições do mesmo, foram construídos os seguintes diagramas de casos de uso, referentes aos módulos componentes do subsistema.

O diagrama exibido pela Figura 21 nos apresenta o comportamento relacional do módulo de gerenciamento com os seus atores, o subsistema iVoice e o módulo servidor.

Figura 21 Casos de Uso - Módulo de Gerência

O comportamento do módulo servidor em relação aos demais módulos do subsistema e ao servidor Web é visualizado no diagrama de casos de uso mostrado pela Figura 22.

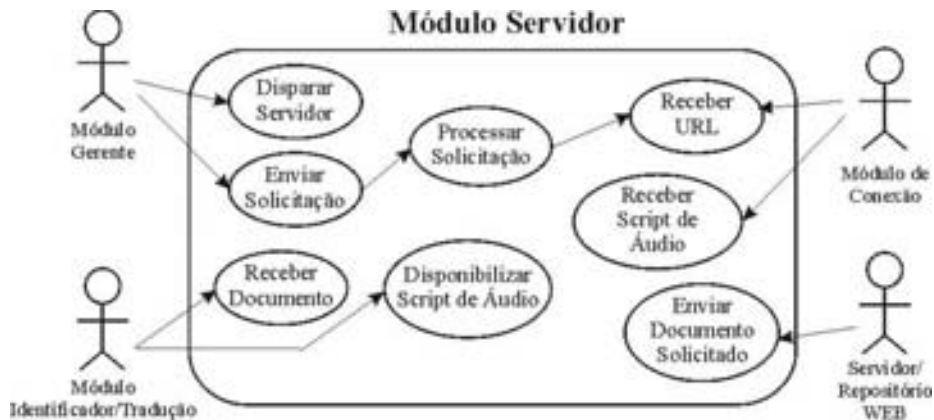



Figura 22 Casos de Uso - Módulo Servidor

Na Figura 23, pode-se visualizar o diagrama de casos de uso referente ao módulo de conexão e seus respectivos atores: iVoice, módulo servidor e servidores/repositórios Web.

Figura 23 Casos de Uso - Módulo de Conexão

Finalmente, encerrando a apresentação dos diagramas de casos de uso, pode ser visualizado na Figura 24, o diagrama comportamental do módulo de tradução em relação ao seu único ator, o módulo servidor.

Figura 24 Casos de Uso - Módulo de Tradução

5.1.2 Diagrama de Atividades

Os diagramas de atividades, segundo BOOCH, RUMBAUGH, e JACOBSON [17], são empregados com a finalidade de se modelar aspectos dinâmicos do sistema. Neste projeto, a utilização de tal diagrama justificou-se em função das necessidades de se visualizar, modelar, especificar e documentar as atividades e relacionamentos envolvidos no processo computacional, dando-se ênfase ao fluxo de controle de uma atividade para outra.

As atividades envolvidas no funcionamento do tVoice (e seus módulos) e na comunicação com o iVoice e com o servidor/repositório Web são visualizadas na Figura 25.

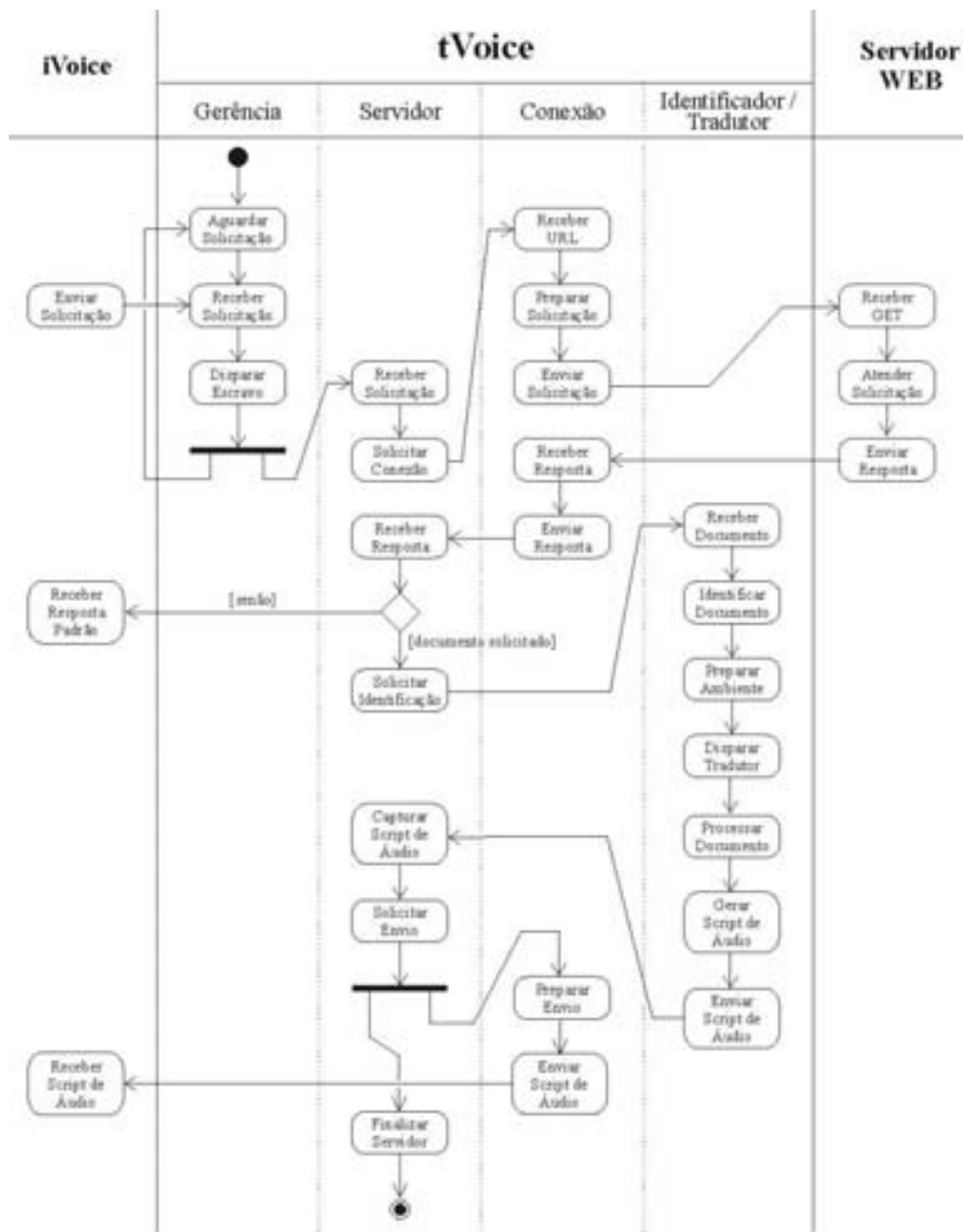



Figura 25 Diagrama de Atividades

Esse diagrama apresenta inicialmente os passos de inicialização do tVoice aguardando por uma solicitação proveniente do subsistema iVoice. Em seguida ao recebimento dessa solicitação, acontece o disparo de dois processos paralelos: o primeiro de retorno do módulo de gerência ao estado de espera e o segundo de instanciação de um novo servidor-escravo para atender, de forma individual, a solicitação recebida.

Imediatamente após esse procedimento, os processos de comunicação com o servidor Web são inicializados, em procurando recuperar o documento solicitado. Se a resposta do servidor for positiva, no sentido de haver encontrado o documento requerido, um novo estágio é disparado; caso contrário, é retornado ao iVoice uma página padrão reportando o erro de inexistência do documento, naquele servidor.

O novo estágio, caso tenha sido encontrado o documento pedido, refere-se aos processos de identificação do documento recebido e do devido tratamento do mesmo, em busca do processamento de extração das informações textuais contidas.

Ao final desses processos, tendo sido gerado o script de áudio, o mesmo é enviado ao subsistema de interface, finalizando a instanciação do servidor-escravo, inicialmente disparado pelo módulo de gerência.

5.1.3 Diagrama de Seqüências

Um diagrama de seqüências é um diagrama de interação que enfatiza a ordenação temporal das mensagens trocadas entre componentes formadores de um sistema. Assim sendo, tal diagrama fornece uma visão mais dinâmica do processo envolvido no funcionamento do mesmo [17].

A Figura 26A e B apresenta diagramas de seqüência referentes aos módulos que compõem o tVoice, relacionando-os com o iVoice e o Servidor Web, quando de uma solicitação bem sucedida e uma mal sucedida, respectivamente.

Figura 26A Diagrama de Seqüências - Solicitação bem sucedida

Figura 26B Diagrama de Seqüências - Solicitação mal sucedida

Os diagramas apresentados mostram a ação temporal de ativação de cada módulo do subsistema tVoice, durante todo o processo de funcionamento do mesmo, no período de recebimento e processamento de uma solicitação.

Logo, tendo ambos os subsistemas inicializados e já em execução, uma mensagem de solicitação de documento é disparada em direção ao módulo de gerência do tVoice, dando início ao processo de recuperação e tratamento do requerido documento.

Após o recebimento de uma solicitação, o módulo de gerência retorna ao estado de espera e instancia um servidor-escravo, dedicado a atender àquele pedido. Esse novo módulo instanciado imediatamente dispara uma conexão com o servidor/repositório Web, a procura do documento solicitado. Sendo encontrado e recebido o documento requisitado, o mesmo é enviado ao módulo identificador/tradutor, que efetua as primeiras análises, preparando o ambiente para o processo de extração da informação, a ser efetuado, imediatamente em seguida.

Processado e gerado o script de áudio, o mesmo é enviado pelo módulo servidor ao subsistema de interface, através do módulo de conexão, finalizando todo o processo.

5.1.4 Diagrama de Classes

A importância dos diagramas de classes para o projeto se dá em função da visão estática do sistema em classes, associadas aos seus relacionamentos e permitindo a

modelagem e uma melhor aproximação do desenvolvimento orientado a objetos [17], metodologia de desenvolvimento escolhida para a implementação.

Através do diagrama apresentado pela Figura 27, utilizando ainda a notação UML, podemos observar as classes que compõem o subsistema tVoice e seus relacionamentos de agregação, especialização, dependências e associações, bem como a relação de seus respectivos atributos e métodos.

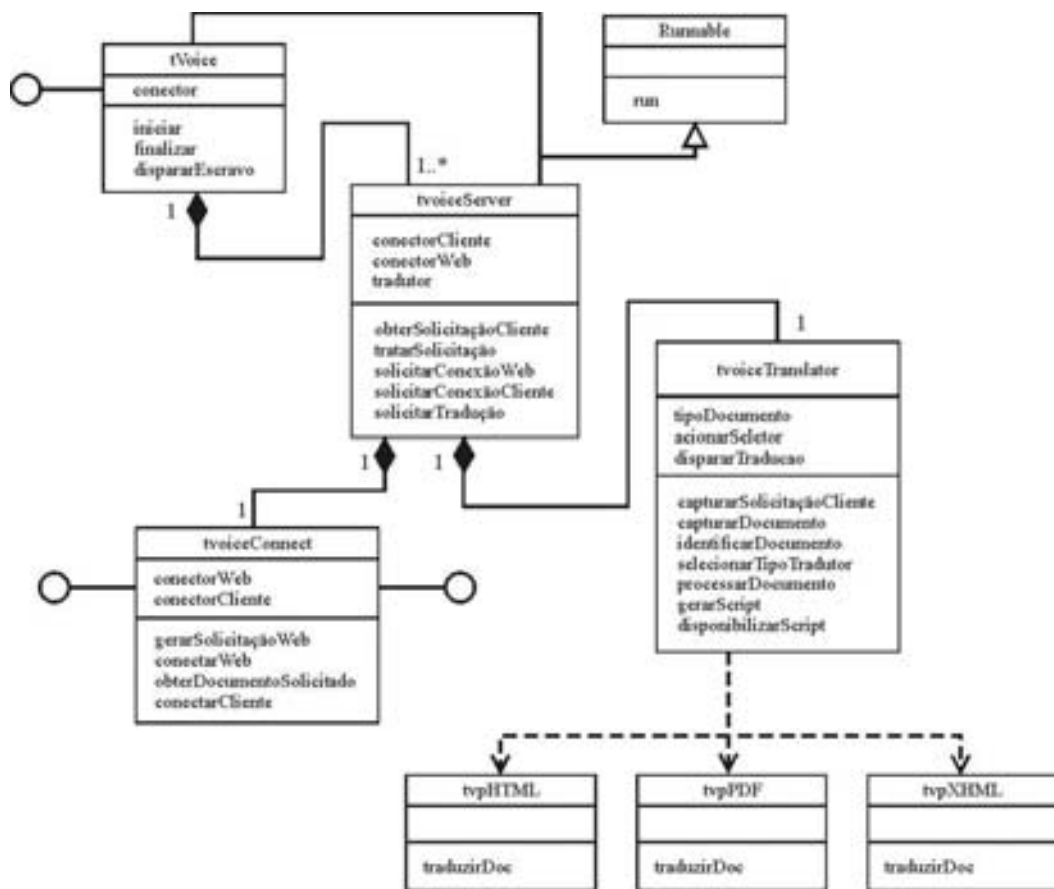



Figura 27 Diagrama de Classes

Deste ponto em diante, aplicando fonte itálico aos nomes das classes, para que não haja confusão ao se falar do subsistema e seus módulos, podemos explicar o diagrama apresentado anteriormente.

Tal diagrama proporciona, como dito, a visualização das classes do subsistema de tradução, mostrando primeiramente a implementação da função *run* da classe abstrata Runnable, inerente a biblioteca padrão do Java, pelas classes *tVoice* e *tvoiceServer*.

Seguindo uma visualização vertical do diagrama, de cima para baixo, são apresentadas as agregações da classe *tvoiceServer*, em relação à classe *tVoice*, e as das classes *tvoiceConnect* e *tvoiceTranslator*, em relação à classe *tvoiceServer*.

O diagrama ainda nos apresenta a existência da interface de comunicação da classe *tVoice*, utilizada no processamento junto ao subsistema de interface do VoiceProxy, bem como, as interfaces de comunicação da classe *tvoiceConnect*, utilizadas na conversação com o servidor/repositório Web e também com o subsistema de interface, cliente do subsistema de tradução.

Finalmente, são visualizadas as dependências das classes de *parsers* em relação à classe *tvoiceTranslator,* utilizadas no processamento dos documentos para a realização da tradução dos mesmos.

5.2 Implementação

Previamente, optamos por utilizar a linguagem de programação Java, versão 1.3.1, na fase de implementação do subsistema de tradução, em função de algumas de suas características, tais como sua portabilidade, ampla gama de componentes já implementados para comunicação na Internet e, por fim, sua perfeita integração com as ferramentas de software utilizadas na implementação do módulo Identificador/Tradutor e seus componentes.

O subsistema de tradução foi desenvolvido sobre a plataforma operacional Windows, porém, em função da linguagem de programação Java ser uma linguagem portável, testes realizados revelaram um bom desempenho funcional também na plataforma operacional Linux, conferindo ao mesmo uma maior usabilidade e portabilidade.

Assim como citado anteriormente, utilizou-se o paradigma de programação orientado a objetos para a implementação dos módulos que compõem o tVoice, a saber os módulos de gerenciamento, servidor-escravo, conexão e identificação/tradução, sendo definidas as seguintes classes: *tVoice*, *tvoiceServer*, *tvoiceConnect*, *tvoiceTranslator*, respectivamente, apresentadas pelo diagrama de classes da Figura 27.

As seções seguintes apresentam mais detalhadamente as funcionalidades e características das classes utilizadas na implementação do subsistema de tradução.

5.2.1 A Classe tVoice

Como apresentado, a classe *tVoice* representa a implementação do Módulo Gerente. Ela possui um atributo e três métodos associados, desempenhando a função básica de receber, disparar e repassar solicitações a instâncias da classe *tvoiceServer*.

O atributo denominado "conector", como visto no diagrama de classes, é uma instância da classe Java java.net.ServerSocket e é responsável pelo aguardo de uma solicitação, escutando em uma porta lógica predefinida e configurada pelo administrador do sistema, especificamente escolhida para a realização do atendimento aos clientes.

Utilizando o modelo arquitetural Cliente-Servidor e uma política concorrente de atendimento aos clientes, essa classe possui, dentre seus métodos, o método "dispararEscravo", responsável pela instanciação de uma classe *tvoiceServer* a cada solicitação recebida, proveniente do subsistema de interface do VoiceProxy.

O fato de se agenciar uma nova instância da classe *tvoiceServer* para atender separadamente a cada solicitação garante que qualquer problema ocorrido com um dos servidores escravos não cause reações em cadeia, interferindo no atendimento dispensado a outros clientes, conferindo assim características de confiabilidade ao atendimento.

Ainda no rol de métodos dessa classe, podemos encontrar as funções "inicializar" e "finalizar", tendo como tarefas, logicamente, dar início e parar o processo de atendimento às solicitações, por parte do subsistema de tradução.

A seguir, o trecho de código em Java 1.3.1 nos permite visualizar a codificação funcional base dessa classe.

```
public class tVoice extends javax.swing.JFrame implements Runnable{
...
   porta = Integer.parseInt(tf_porta.getText().trim());
   conector = new java.net.ServerSocket(porta);
...
   while (status){
      java.net.Socket rcv = conector.accept();
```

```
      //disparo de novos servidores-escravos
      new tvoiceServer((Thread.currentThread().MAX_PRIORITY
      - 2),tf_solicitacoes, rcv).iniciar();
   } //fim while

   //finalizaçao de escuta no servidor
   tServer.close();
...
}//fim classe tVoice
```

Uma interface simples foi desenvolvida para facilitar o manuseio e configuração do servidor. Essa pode ser visualizada na Figura 28.

Figura 28 Interface tVoice

5.2.2 A Classe tvoiceServer

Sendo a representação da implementação do módulo Servidor, a classe *tvoiceServer* é composta basicamente por três atributos e cinco métodos, tendo como função principal atender as solicitações do cliente quanto ao agenciamento de todo o processo envolvido na recuperação e tradução dos documentos.

Depois de ser instanciado e inicializado um objeto da classe *tvoiceServer*, o primeiro passo é obter do cliente o endereço (URL) desejado, através da utilização do protocolo HTTP 1.0, para que, assim, o tVoice possa realizar a recuperação do documento, representado pela URL (Uniform Resource Locator), na Internet.

Seguido a esse procedimento e tendo sido recebido do cliente tal informação, a nova instância do *tvoiceServer* instancia uma classe *tvoiceConnect*, para que se processe a recuperação do documento desejado, através de solicitações a um servidor Web.

Dado que uma resposta positiva seja recebida do servidor Web em que o documento solicitado se encontra, isto é, caso o documento pedido seja recebido como resposta do servidor ao qual se fez a solicitação, uma classe *tvoiceTranslator* é instanciada para a realização da verificação do tipo do documento recebido e, assim, seja desencadeado o processo de tradução do mesmo, nessa mesma classe.

Finalmente, tendo o *script de áudio* gerado como resposta ao processo de tradução, o mesmo é enviado através de uma nova instância da classe *tvoiceConnect*, também através do protocolo HTTP 1.0, ao cliente.

As linhas de codificação Java a seguir nos apresentam alguns detalhes da implementação da classe *tvoiceServer*.

```
public class tvoiceServer implements Runnable {
...
   //checagem se endereco foi recebido corretamente
   if (ck_conexao){
...
      conexao = new tvoiceConnect(ponto_conexao);
      conexao.conectar();
      //checagem se houve êxito na tentativa de conexão com servidor
      if (conexao.conectServ){
...
         //recebimento resposta do servidor Web
         doc_recebido = conexao.getResposta();
...
            //instanciacao do modulo de traducao
            tradutor = new tvoiceTranslator(conexao.getEndrc(),
                       doc_recebido);
...
         //checagem de identificacao para geracao script
         if (tradutor.tradStatus){
            doc_script_audio = tradutor.gerarScript();
...
         }else{
```

```
            doc_script_audio = null;
         }//fim if-else
...
      } //fim if conexao.conectServ
...
      //envio de script de audio ao cliente
      conexao2 = new tvoiceConnect(sck_cliente, doc_script_audio);
      conexao2.enviar_script();
...
} //fim classe tvoiceServer
```

5.2.3 A Classe tvoiceConnect

A classe *tvoiceConnect* foi implementada com a finalidade de ser a representação lógica do módulo Conexão, sendo responsável, como já apresentado, por todo o processo de estruturação da comunicação na Internet, via HTTP, com outros servidores Web e com o subsistema de interface.

Depois de instanciada, essa classe se encarrega de montar toda a estrutura necessária para se enviar e receber documentos do tipo MIME através da rede, como citado anteriormente, através do protocolo HTTP 1.0.

O trecho de código apresentado abaixo nos dá uma idéia superficial das funcionalidades gerais dessa classe.

```
public class tvoiceConnect{
...
   //1o. construtor de intanciacao
   public tvoiceConnect(String endrc){
...
            sltConexao = new java.net.Socket(endrc,80);
...
            String cab_env_servidor = "GET " + doc_slt
            + "HTTP/1.0\n"
            + "Accept: *.*\n"
            + "Accept-Language: pt-br\n"
            + "Accept-Encoding: deflate\n"
            + "User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows
               98; Win9x 4.90)\n" +
            + "Host: " + endereco_tratado + "\n" +
            + "Connection: close\n\n";
...
   }//fim 1o. construtor
```

```
   //2o. construtor de instanciacao
   public tvoiceConnect(java.net.Socket sck_cliente, byte[] doc_env){
...
            con_cliente = sck_cliente;
...
            String cab_env_cliente = "HTTP/1.0 200 OK\n"
            + "Content-Type: text/html\n"
            + "Connection: close\n\n";"
...
   }//fim 2o. construtor

}//fim classe tvoiceConnect
```

5.2.4 A Classe tvoiceTranslator

Como implementação do módulo de identificação/tradução, a classe *tvoiceTranslator* é assim denominada em função de seu principal objetivo funcional ser efetuar uma tradução nos documentos solicitados pelo subsistema de interface e encontrados na Web.

Essa classe é responsável também pela extração e geração de um documento chamado de *script de áudio*, onde são armazenadas as informações textuais extraídas dos documentos tratados, acrescidas de marcações (*tags*) de áudio, utilizadas pelo subsistema de interface no processo de síntese de voz, na interação com os usuários.

Antes da fase de extração de informações dos documentos, uma prévia identificação é realizada, utilizando-se a requisição inicialmente recebida pela instância da classe *tvoiceServer*, para em seguida se instanciar a respectiva classe de *parser* que corresponde ao tratamento do tipo de documento a ser processado, seja ele HTML (*HyperText Markup Language*), PDF (*Portable Document Format*) ou XHTML (e*Xtensible HyperText Markup Language*).

Os principais métodos dessa classe são: *selecionarTipoTradutor()*, responsável pela identificação da solicitação e instanciação do determinado tipo de *parser* a ser utilizado, e *gerarScript()*, responsável pelo disparo do processo de tradução propriamente dito.

O fragmento de codificação Java apresentado a seguir nos proporciona uma imagem geral do funcionamento da classe *tvoiceTranslator*.

```
public class tvoiceTranslator {
...
   parserId = selectTradutor();
...
   switch (parserId){
      case thtml:
         tvpHtml phtml = new tvpHtml();
         script = phtml.tradDoc(doc_recebido);
         break;
      case tpdf:
         tvpPdf ppdf = new tvpPdf()
         script = ppdf.tradDoc(doc_recebido);
         break;
      case txhtml:
         tvpHtml pxhtml = new tvpHtml()
         script = pxhtml.tradDoc(doc_recebido);
         break;
   }//fim switch
...
}//fim classe tvoicTranslator
```

As classes apresentadas a seguir são as de *parsers* propriamente ditas, responsáveis pelo processo de tradução realizado nos documentos recebidos da Web, em resposta às solicitações realizadas pelo subsistema de tradução, originárias inicialmente do subsistema de interação.

Destas, apenas características relacionadas à conversão serão apresentadas nas seções seguintes, com a finalidade de um maior esclarecimento do processo.

É válido ressaltar que as marcações (*tags*) de áudio empregadas no processo de tradução dos documentos e geração do script final são as utilizadas no sintetizador IBM ViaVoice TTS Runtime PRO - Release 7 BR, sintetizador escolhido e empregado no processo de interação com o usuário pelo subsistema de interação, o iVoice.

5.2.5 A Classe tvpHtml

Baseando-se no estudo realizado sobre a linguagem de marcação HTML versão 3.2 e também no projeto AHA, apresentado no capítulo 3, desenvolveu-se um

compilador que executasse uma tradução do formato HTML para o *script de áudio* utilizado no processo de síntese de voz para o usuário.

O processo de tradução busca extrair a maior quantidade de informação textual possível, a partir da utilização das próprias *tags* da linguagem fonte e suas funcionalidades.

A versão HTML estudada e tratada pelo *parser* é, como dito, a 3.2, baseando-se na primeira recomendação do W3C [26,28] e no projeto AHA [21].

Para possibilitar a tradução, foram definidas equivalências entre as *tags* HTML reconhecidas e tratadas e as marcações de áudio utilizadas, salientando que nem todas as *tags* que compõem a linguagem HTML são tratadas em nível de tradução para um equivalente em áudio.

Na tabela 5, podemos visualizar as *tags* de áudio utilizadas nesta versão do subsistema juntamente com suas respectivas funções.

Tabela 5 Marcações de áudio utilizadas no tvpHtml

MARCAÇÕES	FUNÇÃO
\Pau=N	Cria uma pausa com N milésimos de segundos de duração.
\xPfl=N	Define a flutuação de tom como N, alterando a entonação com que é pronunciada a palavra.
\Rst	Redefine características originais do locutor selecionado.
\Vce=Speaker=<*locutor*>	Define quem será o locutor, selecionando *nome* dentre as diferentes vozes disponíveis no sintetizador.

5.2.5.1 Tags HTML em Tags de Script de Áudio

Na tabela 6 podemos observar os relacionamentos de *tags* HTML com as *tags* de áudio utilizadas pela classe *tvpHtml*.

Tabela 6 Relacionamentos de tags HTML – tags Áudio

	TAGS HTML	TAGS DE ÁUDIO
TÍTULO	<TITLE>	\xPfl=70\ Título da pagina:
LINKS	<A>	\Vce=Speaker=Cláudia\ Link
	<A> (internos)	\Vce=Speaker=Cláudia\ Link \Mrk=6\ <texto> \Mrk=9\
	<AREA>	\Vce=Speaker=Cláudia\ Link
TEXTO EM DESTAQUE	<B>	\xPfl=70\
	<I>	\xPfl=70\
	<BIG>	\xPfl=70\
	<H1> a <H6>	\Pau=500\ \xPfl=70\
	<U>	\xPfl=70\
	<CODE>	\xPfl=70\
	<CITE>	\xPfl=70\
	<ADDRESS>	\xPfl=70\
IMAGENS	<IMG>	\Pau=500\ Imagem
		\Pau=500\ Imagem sem descrição
LISTAS DE OPÇÕES	<SELECT>	\Pau=500\ Opções de:
	<LI>	\Pau=500\
FIM		\xPfl=70\ Fim do Documento.

Como indicação do fim do reconhecimento de uma *tag* HTML, dentro do *script* gerado, utilizamos a *tag* de áudio **\Rst**, para reconfigurar o sintetizador para as características de voz padrão.

É ainda válido salientar que nem todas as *tags* HTML possuem um correspondente direto no script de áudio, fato este exemplificado pelo não processamento adequado da *tag* <TABLE>, sendo esta tratada como uma busca seqüencial nas linhas da tabela encontrada.

Em específico, a justificativa para o não tratamento adequado dessa *tag* se dá pelo fato de alguns *WebDesigners* fazerem mau uso da mesma na estruturação visual dos documentos, o que, segundo o estudo realizado no projeto AHA e pesquisas Web,

dificultam o tratamento dos documentos nesses casos. Essa má utilização vai de encontro às diretivas de Acessibilidade do W3C, que aconselham a utilização da *tag* TABLE apenas para formatação de tabela de dados.

Infelizmente o processo de tradução em documentos HTML não esta sujeito apenas ao bom funcionamento do tVoice.

No processo de tradução existe uma forte dependência de como cada documento HTML foi desenvolvido, pois essa ferramenta não conseguirá maximizar a quantidade de informações textuais extraídas de páginas que possuam tecnologias do tipo: *applets*, VRML (Virtual Reality Modelling Language), *scripts*, animações (plugins ou figuras animadas) e formulários (gerados em linguagens como: php, jsp, perl, asp), bem como *frames* (ainda não tratados nesta versão do subsistema).

Por esse motivo, cabe aos designers utilizarem as regras de Acessibilidade disponíveis na Internet, para assim tornarem suas páginas mais acessíveis não só para os deficientes, mas para qualquer outro tipo de usuário, como por exemplo, usuários de *browsers* não visuais.

5.2.6 A Classe tvpPdf

Para realizar a tradução de documentos PDF essa classe utiliza um pacote denominado Pj.jar, desenvolvido e disponibilizado gratuitamente pela empresa Etymon Systems em Java, que possui um conjunto simplificado de ferramentas para manipulação e tratamento de documentos PDFs.

Esse pacote possui algumas restrições, que conseqüentemente estendem-se também ao subsistema de tradução.

Para que os documentos possam ser tratados, devem possuir uma ou mais das seguintes características:

1. Se houver codificação, utilizar a ASCII85
2. Se houver compressão, utilizar a JPEG
3. Possuir permissão para leitura
4. Possuir permissão para extração ou cópia de conteúdo

Da estrutura sintática, formadora de um documento PDF, apenas o chamado *Content Stream* é tratado e processado pelo *parser* da classe *tvpPdf*, em busca de conteúdo textual a ser extraído. Isso, baseado na visão de que os outros componentes sintáticos formadores de um documento PDF, segundo o PDF Reference version 1.5 [27], estão associados à formatação do documento. São eles:

- *Objects* – que correspondem aos tipos de objetos básico formadores de um documento.
- *File structure* – que corresponde à estrutura que determina como os objetos estão distribuídos em um documento.
- *Document structure* – que corresponde a como os tipos básicos são organizados para representar componentes de um documento PDF tais como: páginas, fontes, anotações, etc.

Figura 29 Componentes de um documento PDF

Na tabela 7 algumas das conversões feitas pela classe *tvpPdf* na tradução de documentos PDF.

Tabela 7 Relacionamentos de Objetos PDF – tags Áudio

PDF	TAGS DE ÁUDIO
Autor	\Vce=Speaker=Cláudia\ Autor: \xPfl=70\
Assunto	\Vce=Speaker=Cláudia\ Assunto: \xPfl=70\
Palavras Chave	\Vce=Speaker=Cláudia\ Palavras Chave: \xPfl=70\
p/ cada Content Stream	\Pau=500\ \xPfl=70\ <texto>
Fim do documento	\xPfl=70\ Fim do Documento.

Para a conversão dos documentos PDF são utilizadas as mesmas marcações de áudio empregadas na tradução dos documentos HTML, apresentadas na seção anterior, pela tabela 5.

5.2.7 Tratamento de XHTMLs

Atualmente, o subsistema de tradução efetua o tratamento e processamento apenas de documentos XHTML *Strictly Conforming*, isto é, documentos XHTML que possuam apenas tags HTML, baseados no DTD (*Document Type Definitions*) "xhtml1-strict.dtd" [25], identificado em um documento como apresentado a seguir:

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Dessa forma, pequenas alterações feitas na classe de *parser tvpHtml* possibilitam sua utilização também no tratamento de documentos XHTML.

Assim sendo, os mesmos relacionamentos observados na tabela 6 são estendidos também para a tradução dos documentos XHTML.

Um exemplo básico de um documento XHTML *Strictly Conforming*, hoje tratado pelo tVoice, pode ser visualizado a seguir [25]:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Documento XHTML Teste</title>
  </head>
  <body>
    <p>Exemplo de link <a href="http://www.teste.com.br">Teste</a>.</p>
  </body>
</html>
```

5.3 Testes e resultados obtidos

Os testes realizados com o subsistema desenvolvido tiveram o objetivo de avaliar aspectos gerais relacionados aos sistemas distribuídos, levando em conta características tais como Transparência, Confiabilidade e Desempenho, relacionando-os à concorrência e paralelismo no atendimento.

Para realização dos testes foram utilizados *browsers* visuais, GET Linux e uma ferramenta, denominada SimGets, desenvolvida também em Java, para tentar simular solicitações oriundas do subsistema de interface, usando o protocolo HTTP 1.0. Tal fato se deu em função do subsistema de interface ainda se encontrar em fase de desenvolvimento.

O SimGets é capaz de disparar *n* solicitações quase simultaneamente, procurando simular a concorrência no atendimento a várias solicitações feitas através da Internet.

Para esses testes foram utilizadas máquinas de mesma configuração, com processadores de 1.1MHz, 128Mb de memória RAM e placas de rede convencionais de 10/100Mbps, sem levar em conta características tais como arquitetura e velocidade de barramento das placas-mãe utilizadas. Estes equipamentos estão situados nas redes do laboratório do mestrado do DIMAp (Departamento de Informática e Matemática Aplicada da UFRN) e laboratório NatalNet - UFRN.

Em uma delas, escolhida como máquina Servidor e de endereço IP 10.9.98.26, onde toda a estrutura de comunicação com a Internet estava já configurada e em funcionamento, o tVoice foi instalado e adequadamente configurado para escutar e aguardar por solicitações na porta de serviço 7000.

Os documentos utilizados nos testes foram publicados no servidor Web do laboratório Natalnet, também situado no DIMAp - UFRN.

5.3.1 Testes com browsers

Utilizando-se dos *browsers* InternetExplorer 5.50, Mozilla 4 e Lynx 3.2 (*browser* não visual) foram simuladas solicitações de diferentes documentos via HTTP e

para isso os seguintes endereços de documentos foram digitados na barra de endereços, como visto na tabela 8 a seguir.

Tabela 8 Solicitações via browser

ID	ENDEREÇOS DOS DOCUMENTOS	TAM. ARQUIVO	TIPOS
1	http://10.9.98.26:7000/=www.natalnet.br/~ihsg/teste_html.html	146 Kb	HTML
2	http://10.9.98.26:7000/=www.natalnet.br/~ihsg/teste_pdf.pdf	104 Kb	PDF
3	http://10.9.98.26:7000/= www.natalnet.br/~ihsg/teste_xhtml.html	10 Kb	XHTML

Cada solicitação foi realizada 3 vezes, tomando ao final o tempo médio de atendimento, desde do disparo até o recebimento do script de áudio gerado, o que possibilitou a construção da seguinte tabela, relacionando os tempos médios aproximados de cada atendimento.

Tabela 9 Resultados – testes browsers

ID	TIPO	TEMPO ≅
1	HTML	1,5”
2	PDF	2”
3	XHTML	1,2”

Durante esses testes o subsistema comportou-se bem, dentro do esperado, em relação aos aspectos de transparência e confiabilidade, realizando as tarefas para as quais foi desenvolvido, dentro dos requisitos de suas limitações.

O Anexo I contém alguns fragmentos dos documentos utilizados nos testes com *browsers*, juntamente com seus scripts de áudio, correspondentemente gerados.

5.3.2 Testes com a ferramenta SimGets

Como anteriormente citado, a ferramenta denominada de SimGets foi desenvolvida para ser capaz de simular *n* solicitações GET, quase que simultaneamente, com a finalidade de testar, no tVoice, a característica desempenho, quando da concorrência no atendimento aos clientes. Sua interface é apresentada pela Figura 30, abaixo.

Figura 30 Interface SimGets

Utilizando-se da ferramenta e dos mesmos arquivos apresentados na tabela 8, foram feitos testes com aumentos gradativos do número de solicitações. O que nos proporcionou a obtenção dos seguintes valores aproximados, apresentados pela tabela 10.

Tabela 10 Resultados – testes SimGets

ID	TIPO	NO. SOLICITAÇÕES	TEMPO ≅
1	HTML	10	1’ 53”
		40	5’ 12”
		100	14’ 38”
2	PDF	10	2’ 45”
		40	6’ 40”
		100	15’ 17”
3	XHTML	10	38”
		40	3’ 43”
		100	9’ 51”

Observando os resultados obtidos dos testes realizados com a ferramenta, foi possível se chegar a algumas constatações, tais como:

- Tempo de atendimento muito elevado para os padrões convencionais de atendimento Web para os tipos de documentos tratados.
- Quantidade elevada de perdas de conexões com o servidor, a partir de 40 solicitações. Cerca de 20% das conexões com o servidor tVoice eram encerradas, gerando perda de dados.

- Sensação de serialização no atendimento aos clientes simulados pelo SimGets.

Comparando os resultados dos testes realizados com *browsers* e com os com o SimGets, verificou-se uma grande disparidade, o que nos levou a refletir sobre os experimentos e levantar as possíveis causas para o acontecido.

Alguns pontos foram levantados e relacionados às possíveis causas dos problemas de conexão e do retardo no atendimento as solicitações simuladas. São eles:

1. A utilização de uma única máquina, com uma única placa de rede e um único link com o servidor, para simular inúmeras solicitações por vez, causando o efeito gargalo, tanto na saída das solicitações quanto no recebimento das respostas.
2. Possíveis conflitos de sockets no disparo das solicitações.
3. Retardo causado pela constante paginação de memória na máquina cliente.

Diante dos fatos, concluiu-se que a utilização da ferramenta SimGets não foi a maneira mais adequada para auxiliar na averiguação do desempenho do tVoice quanto ao atendimento concorrente aos clientes, uma vez que a mesma, por limitações físicas, não era capaz de realizar a tarefa de simulação com perfeição uma situação real.

Tal fato desencadeou a realização de um terceiro e último experimento utilizando-se o sistema Linux e a ferramenta GET 1.4, distribuída juntamente com esse sistema operacional.

5.3.3 Testes com GET Linux

Para esses experimentos, realizados no próprio ambiente Linux, foi utilizada em cada máquina, num total de 8 (oito) estações, além da aplicação GET, a ferramenta Crontab, responsável pelo agendamento de tarefas a serem executadas no sistema linux, por um determinado período de tempo.

A seguir é apresentado o arquivo de configuração desse agendador de tarefas, habilitando o mesmo a executar dois scripts a cada minuto, indeterminadamente, em cada uma das máquinas, até que o serviço de agendamento fosse desabilitado,

direcionando ainda os resultados dos mesmos para dois arquivos de texto, também situados nas referidas máquinas utilizadas nos testes.

ARQUIVO DE CONFIGURAÇÃO CRONTAB (COMANDO: Crontab –e)
`* * * * * exec script1 >> result1`
`* * * * * exec script2 >> result2`

Subseqüentemente, temos os scripts utilizados e executados em cada máquina, disparando em seqüência os comandos de apresentação da data e hora, marcando início e fim da solicitação, e o comando GET, simulando as solicitações HTTP, propriamente ditas.

O comando GET, como dito, é distribuído livremente, juntamente com o sistema operacional Linux, e é utilizado para efetuar solicitações HTTP a qualquer servidor Web.

SCRIPT 1

```
date
GET http://10.9.98.26:7000/=www.natalnet.br/~ihsg/teste_html.html -d
date
```

SCRIPT 2

```
date
GET http://10.9.98.26:7000/=www.natalnet.br/~ihsg/teste_pdf.pdf -d
date
```

Para que se pudesse identificar a qual estação cliente pertencia os resultados gerados, em cada máquina, os arquivos de texto com os resultados foram renomeados seguindo o seguinte padrão:

NOMES DE ARQUIVOS DE RESULTADOS:
rst_[nome_máquina]1 – gerado pela execução do script 1
rst_[nome_máquina]2 – gerado pela execução do script 2

Fragmentos dos resultados obtidos através da execução dos scripts, no período de tempo de 2 (duas) horas, com disparos simultâneos oriundos de 8 (oito) máquinas diferentes, gerando o número total de quase 1000 solicitações no tVoice nesse mesmo intervalo de tempo, podem ser observados no Anexo II deste trabalho.

Desses fragmentos podemos extrair os dados apresentados pela tabela 11, referentes aos tempos médios de atendimento às solicitações em cada máquina utilizada.

Tabela 11 Resultados – testes SimGets

MÁQUINA	TIPO	TEMPO MÉIDO DE ATENDIMENTO
10.9.98.17	HTML	1”
	PDF	2”
10.258.0.16	HTML	4”
	PDF	5”
10.9.98.18	HTML	2”
	PDF	3”
10.9.98.26	HTML	2”
	PDF	2”
10.9.98.15	HTML	1”
	PDF	2”
10.9.98.16	HTML	2”
	PDF	2”
10.9.98.22	HTML	2”
	PDF	2”
10.9.98.27	HTML	2”
	PDF	2”

Uma importante observação a ser considerada em relação aos resultados foi a não sincronização dos relógios das máquinas, uma vez que se necessita de permissões de Administrador (root) para realizar tais alterações. Tal fator justificou a não compatibilidade de horários nos *logs*, o que não interferiu na execução e geração dos mesmos.

Diante dos resultados obtidos com esses testes, verificou-se que o tVoice, agora funcionando sobre uma arquiterura cliente-servidor, comportou-se dentro do desejado quanto ao atendimento concorrente a soliticitações, conseguindo atingir um desempenho suficientemente regular e satisfatório.

6. Conclusões e Perspectivas Futuras

Este trabalho apresentou todo o processo de projeto e modelagem, bem como implementação, da nova versão do tVoice - subsistema de tradução do VoiceProxy - sistema que auxilia o acesso de pessoas portadoras de deficiência visual à Internet, promovendo a Acessibilidade e simplificando a utilização do maior repositório de informação existente, a Web.

Tendo seu funcionamento agora distribuído, o tVoice permite uma maior flexibilidade de uso tanto para o subsistema de interface utilizado quanto para o usuário alvo, eximindo-os da necessidade de instalação e configuração de sistemas de softwares adicionais, permitindo o acesso de qualquer localidade e diminuindo a carga de processamento na máquina do usuário.

Por utilizar diretivas básicas do protocolo padrão da Internet, o HTTP, em sua comunicação, o subsistema de tradução possibilitará a facilitação de seu uso também por outros subsistemas de interface.

Sabemos que o tVoice ainda necessita de melhorias computacionais, como por exemplo dar suporte a páginas que contenham frames, utilizar CSS (Cascading Style Sheets) na tradução das páginas, permitir o processamento de PDFs com vários tipos de compressão e trabalhar com documentos XHTML utilizando outros *namespaces*, permitindo assim, que outros tipos de conteúdo sejam traduzidos.

Apesar disso, acreditamos que não só a escada do conhecimento está sendo galgada, mas uma base firme foi desenvolvida para futuros trabalhos que tentem minimizar as barreiras encontradas pelas pessoas portadoras de deficiência.

A modularização do tVoice antevê uma estrutura para dar suporte às seguintes perspectivas:

- Atender e trabalhar com outros tipos de sistemas de interface, produzindo scripts de áudio que possam melhor se adequar aos requisitos impostos pelos mesmos. Por exemplo sistemas de telefonia celular.

- Ampliar gradualmente a quantidade de documentos tratados e processados, permitindo a expansão da gama de informações disponibilizadas aos portadores de necessidades especiais;
- Por fim, tornar-se referência dentro do âmbito computacional da Acessibilidade, propagando a conscientização de um ambiente Web para todos.

6.1.1 Comparação com trabalhos correlatos

Considerando as características de trabalhos correlatos, apresentadas anteriormente, podemos realizar uma breve análise comparativa, relacionando-os com o trabalho desta dissertação, nos seguintes tópicos:

1. Interação com o usuário

Relacionado a questão interação com o usuário, os trabalhos apresentados possuem um sistema acoplado de interação, através da utilização de teclado, mouse e caixas de som, enquanto que o tVoice depende totalmente do subsistema de interação iVoice para permitir a interação através da síntese e reconhecimento de voz como maneira alternativa, além da utilização dos dispositivos padrão anteriormente citados.

Faz-se necessário ressalvar o fato de que o subsistema desenvolvido, objeto alvo deste trabalho, juntamente com o iVoice, compõe o sistema VoiceProxy - sistema de auxílio ànavegação Web para portadores de necessidades especiais.

2. Instalação e utilização do usuário

Para que o usuário utilize um dos sistemas anteriormente apresentados, no capítulo 3 deste trabalho, é necessário que haja a devida instalação dos mesmos em máquina local, podendo ainda demandar a presença de um outro sistema denominado leitor de telas, para permitir o acesso às informações.

Já no sistema VoiceProxy, o usuário precisará ter instalado em sua máquina apenas um *browser* visual, do tipo Internet Explorer, Netscape, Mozilla, etc., pelo qual acessará o sistema. Isso se dá em função de ambos os subsistemas que o compõem funcionarem semelhantemente a servidores, em máquinas dispersas na

Internet, recebendo solicitações via Web, proporcionando um atendimento independente da localidade momentânea do usuário e sistema operacional da máquina do mesmo.

3. Tipos de documentos tratados

O tVoice - subsistema de processamento de documentos do VoiceProxy - permitirá que o usuário utilize mais de um tipo ou formato de documentos disponibilizados na Web, a saber HTML, XHTML e PDF, tendo ainda a perspectiva de tornar-se expansível quanto ao número desses. Comparando com os sistemas associados aos trabalhos correlatos, o usuário tem acesso a informações extraídas apenas de um determinado tipo ou formato de documento.

4. Capacidade de atendimento aos usuários

Quanto a este aspecto, os sistemas apresentados no capítulo 3, em função de serem sistemas de execução local, exceto sistema do modelo WAB, não permitem uso simultâneo de mais de um usuário, fato não comutado pelo VoiceProxy, uma vez que, em seus subsistemas formadores, o modelo arquitetural Cliente-Servidor concorrente é utilizado.

Referências Bibliográficas

[01] TANENBAUM, Andrew S. Distributed Operating Systems. Prentice Hall. 1995.

[02] KIRNER, Claudio e MENDES, Sueli B. T. Sistemas Operacionais Distribuídos - Aspectos Gerais e Análise de sua Estrutura. Editora Campus. 1988.

[03] ALBUQUERQUE, Fernando. TCP/IP Internet: Programação de Sistemas Distribuídos. Axcel Books. 2001.

[04] Tecnologia de Sistemas Distribuídos. mega.ist.utl.pt/~ic-sod. Acessado em Outubro de 2003.

[05] SOARES, Luiz Fernando; LEMOS, Guido e COLCHER, Sérgio. Redes de Computadores - Das LANs, MANs e WANs às Redes ATM. Editora Campus. 1995.

[06] TANENBAUM, Andrew S. Redes de Computadores. Editora Campus. 2003.

[07] MORAES, Alexandre Fernandes e CIRONE, ANTONIO CARLOS. Redes de Computadores: Da Ethernet àInternet. Érica. 2003.

[08] RTF1945 - Especificação HTTP 1.0. www.faqs.org/rfcs/rfc1945.html. Acessado em Novembro de 2003.

[09] Sobre o protocolo HTTP. ltodi.est.ips.pt/Leonardo/ci/MaterialApoio.htm. Acessado em Outrubro de 2003.

[10] AHO, Alfred V.; SETHI, Ravi e ULLMAN, Jeffrey D. Compilers - Principles, Techniques and Tools. Addison Wesley. USA. 1986.

[11] SETZER, Waldemar W. e MELO, Inês S. H. de. A construção de um Compilador. Editora Campus. 1989.

[12] CHEZZI, Carlo e JAZAYERI, Mehdi. Conceitos de Linguagens de Programação. Editora Campus. 1985.

[13] HOLMES, J. Building Your Own Compiler with C++. Prentice Hall. USA. 1995.

[14] JavaCC Documentation. www.cin.ufpe.br/~java/docs/JavaCC/doc/DOC. Acessado em Novembro de 2003.

[15] ELDER, Édipo e PEREIRA, Samara. CL Compiler/CL Editor - Um compilador e editor Java para a linguagem CL. Manual CL Compiler. 2000.

[16] GOMES, Ítalo; TEIXEIRA, Wander; LEMOS, Guido e TAVARES, Tatiana. VoiceProxy: Uma ferramenta de auxílio à navegação na Internet para deficientes visuais. SBMÍDIA'2002. 2002.

[17] BOOCH, Grady; RUMBAUGH, James e JACOBSON, Ivar. UML - Guia do Usuário. Editora Campus. 2000.

[18] BECK, Leland L. Desenvolvimento de Software Básico. Editora Campus. 1994.

[19] Acessibilidade Virtual - Governo Eletrônico. www.governoeletronico.e.gov.br. Acessados em Setembro de 2003.

[20] SONZA, Andréa Poletto e SANTAROSA, Lucila M. C. Ambientes Digitais Virtuais: Acessibilidade aos Deficientes Visuais.
www.cinted.ufrgs.br/renote/fev2003/artigos/andréa_ambientes.pdf. Acessado em Novembro de 2003.

[21] Projeto AHA - Audio HTML Access.
decweb.ethz.ch/WWW6/Technical/Paper296/Paper296.html. Acessado em Novembro de 2003.

[22] WAB - Web Access for Blind users.
www.perrochon.com/archiv/pubs/96sigcaph/index.html.
citesseer.nj.nec.com/kennel96wab.html. Acessados em Novembro de 2003.

[23] ROLLINS, Sami. Audio XML: Aural Interaction with XML Documents. University of California, Santa Barbara. 2000.

[24] GRUNE, Dick; BAL, Henry E. e JACOBS, Ceriel J. H. Projeto Moderno de Compiladores: Implementação a Aplicações. Editora Campus. 2002

[25] XHTML1.0 The Extensible HyperText Markup Language (Second Edition). A Reformulation of HTML 4 in XML 1.0. W3C Recommendation. 26 January 2000, revised 1 August 2002.

[26] Web Access Iniciative (WAI). www.w3c.org/WAI. Acessado em Novembro de 2003.

[27] PDF Reference. Version 1.5. Adobe Portable Document Format. Fourth edition. Adobe Systems Incorporated.

[28] HTML 3.2 Reference Specification. W3C Recommendation. January. 1997.

[29] IBM ViaVoice Outloud API Reference, Version 5.0. November. 1999.

[30] APPEL, Andrew W. Modern Compiler Implementation in Java. Cambridge University Press. New York. 1998.

ANEXO I

Fragmentos de documentos tratados

1o. Fragmento – Documento HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html lang="pt">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Directivas para a Acessibilidade do Conteúdo da Web 1.0</title>
<!-- Changed by: Ian B. Jacobs,  7-Dec-1998 -->
...
<BODY bgcolor="#D1D2D3";>
<P>Este documento é uma versão traduzida de
<A href="http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505">Web Content
Accessibility Guidelines 1.0</A>, do <A href="http://www.w3.org">W3C</A>,
podendo conter erros de tradução. A versão normativa, no idioma inglês,
pode ser encontrada no endereço:</P>
<BLOCKQUOTE><P><A href="http://www.w3.org/TR/WAI-WEBCONTENT">
http://www.w3.org/TR/WAI-WEBCONTENT
</A></P></BLOCKQUOTE>
...
<p>Muita gente não faz idéia do que é, nem que importância possa ter, a
temática da acessibilidade associada à concepção de páginas para a Web.
Pede-se, pois, ao leitor que pense que há muitos utilizadores que actuam
em contextos muito diferentes do seu. Referimo-nos a utilizadores que
podem estar numa das seguintes situações:</p>
<UL>
   <LI>Não ter a capacidade de ver, ouvir ou deslocar-se, ou que podem ter
grandes dificuldades, quando não mesmo a impossibilidade, de interpretar
determinados tipos de informações.
   <LI>Ter dificuldade em ler ou compreender textos.
   <LI>Não ter um teclado ou rato, ou não ser capazes de os utilizar.
   <LI>Ter um ecrã que apenas apresenta texto, um ecrã de dimensões
reduzidas ou uma ligação à Internet muito lenta.
   <LI>Não falar ou compreender fluentemente a língua em que o documento
foi escrito.
   <LI>Ter os olhos, os ouvidos ou as mãos ocupados ou de outra forma
solicitados (por ex., ao volante a caminho do emprego ou a trabalhar num
```

```
ambiente barulhento).
   <LI>Ter uma versão muito antiga de um navegador, um navegador
completamente diferente dos habituais, um navegador por voz, ou um sistema
operativo menos vulgarizado.</LI>
</UL>
<P>Os criadores de conteúdo têm de levar em conta estas diferentes
situações, ao conceberem uma página para a Web. Embora haja uma
multiplicidade de situações, cada projecto de página, para ser
verdadeiramente potenciador da acessibilidade, tem de dar resposta a
vários grupos de incapacidade ou deficiência em simultâneo e, por
extensão, ao universo dos utilizadores da Web. Assim, por exemplo, através
da utilização de <A href="#style-sheet"  title="Definition of style
sheets"><span class="dfn-instance">folhas de estilo</span></A> para
controlo de tipos de letra e para eliminação do elemento FONT, os autores
de páginas em  HTML obtêm um maior domínio sobre as páginas que criam,
tornam-nas mais acessíveis a pessoas com problemas de visão e, através da
partilha de folhas de estilo, reduzem os tempos de transferência de
páginas, para benefício da totalidade dos utilizadores.
...
</body></html>
```

1o. Fragmento – (HTML) Script de Áudio gerado pelo tVoice.

```
\xPfl=70\ Título da página: \Pau=200\ Directivas para a Acessibilidade do
Conteúdo da Web 1.0 \Rst\ \Pau=200\ Este documento é uma versão traduzida
de \Rst\ \Vce=Speaker=Cláudia\ Link  \Pau=200\ Web Content Accessibility
Guidelines 1.0 \Rst\ \Pau=200\ , do \Rst\ \Vce=Speaker=Cláudia\ Link
\Pau=200\ W3C \Rst\ \Pau=200\, podendo conter erros de tradução. A versão
normativa, no idioma inglês, pode ser encontrada no endereço: \Rst\
\Vce=Speaker=Cláudia\ Link  \Pau=200\ http://www.w3.org/TR/WAI-WEBCONTENT
\Rst\
...
\Pau=200\ Muita gente não faz ideia do que é, nem que importância possa
ter, a temática da acessibilidade associada à concepção de páginas para a
Web. Pede-se, pois, ao leitor que pense que há muitos utilizadores que
actuam em contextos muito diferentes do seu. Referimo-nos a utilizadores
que podem estar numa das seguintes situações: \Rst\ \Pau=500\ Não ter a
capacidade de ver, ouvir ou deslocar-se, ou que podem ter grandes
dificuldades, quando não mesmo a impossibilidade, de interpretar
determinados tipos de informações. \Rst\
```

```
 \Pau=500\ Ter dificuldade em ler ou compreender textos. \Rst\
 \Pau=500\ Não ter um teclado ou rato, ou não ser capazes de os utilizar.
\Rst\
 \Pau=500\ Ter um ecrã que apenas apresenta texto, um ecrã de dimensões
reduzidas ou uma ligação à Internet muito lenta.\Rst\
 \Pau=500\  \Pau=200\ Não falar ou compreender fluentemente a língua em
que o documento foi escrito. \Rst\
 \Pau=500\  \Pau=200\ Ter os olhos, os ouvidos ou as mãos ocupados ou de
outra forma solicitados (por ex., ao volante a caminho do emprego ou a
trabalhar num ambiente barulhento). \Rst\
 \Pau=500\  \Pau=200\ Ter uma versão muito antiga de um navegador, um
navegador completamente diferente dos habituais, um navegador por voz, ou
um sistema operativo menos vulgarizado. \Rst\
 \Pau=200\ Os criadores de conteúdo têm de levar em conta estas diferentes
situações, ao conceberem uma página para a Web. Embora haja uma
multiplicidade de situações, cada projecto de página, para ser
verdadeiramente potenciador da acessibilidade, tem de dar resposta a
vários grupos de incapacidade ou deficiência em simultâneo e, por
extensão, ao universo dos utilizadores da Web. Assim, por exemplo, através
da utilização de \Rst\ \Vce=Speaker=Cláudia\ Link \Pau=200\ folhas de
estilo \Rst\ \Pau=200\  para controlo de tipos de letra e para eliminação
do elemento FONT, os autores de páginas em  HTML obtêm um maior domínio
sobre as páginas que criam, tornam-nas mais acessíveis a pessoas com
problemas de visão e, através da partilha de folhas de estilo, reduzem os
tempos de transferência de páginas, para benefício da totalidade dos
utilizadores. \Rst\
\xPfl=70\ Fim do Documento.
```

2o. Fragmento – Documento PDF

```
%PDF-1.2
%âãÏÓ

10 0 obj
<<
/Length 11 0 R
/Filter /FlateDecode
>>
stream
H‰ VM Û6 ý þ sL [!%Q¢ )6A -Z Fs¨{ %z—KÔŠ–“æ×÷‘¢veoâ~À€A‘CræÍ›7üa»â
K˜¤R²¤*^w+Fþ7ÜÓêÝGF¼Jª ó‡
*j,Òö3½É"·´}\q‘ð4ÌÝ…•'˜L^&"pE4IX Îf\Nv¿ÐoÆ t«èwkjýë`¿ü O—R|ïô`±ÉŠ4ÉiÃ‹D
³w9q k¢×<]š3XŠiîn ËD’Îþ½L‰9ª ?£Œ—W2ç³Óý`õÉ.œ& Aµm-ÖLW›^ ©Õ'µ¦F»F" •_iµ
 õàvY Ù°ª»³= M£ G¤ŒñuŒ¯d lÃóDÊgw˜ ÁŸ9aÑxÓZœIŠÞ×Ú9³7G ‡[ Ý  K %Õî>4}Ò{úƒ
•&´Å·ê°ÖšÎ´æ«Â  öj ¼" tW[lWÆ½>Z aœà¸ ®÷S¤Y0L ^…!L }0uðÂÑÙ  ã¨ƒkðyJ9(W
/Ø#Â|>„É˜{g  Ð1G¦;Ø¡Uø(wY!pfc\o» ý×àµ,÷ç]&8Ö †-Ç ·[G?n•þiM-\$ßx
{HÝ q Ý»JHÎDôQ¤BDq§ðajÒ¸eÐ òA×&Üî]9Û¯ YZïÝÉ6aÖ •ðz ŠÀÉÞ'ÕØAï; RëÓŠÔ¸  Ç
¿5µøÿ/%É •Þ0 3¦ÓqH j G¿ñU®b] "‘ùUa^#5Ù,•ú×õÆÊ*Ÿ      >qì‡ö{õs#r ²$—
‹JÎ¢ü¤IUæÑ à -@ â ´õ ÏÚ …¿ð0¢’m§¦jZ f^9ßfLÅ}¼× H'5Å
Cl0 ;©#²M ò¤ÈùÓh0ÄÄèFÿ= ¨…Ó->>(ø:9æ
pzv!‘©  `Q  Ó…<†ú ÅR×÷ƒýìô0§å¹ /u5„ ² ±Dî5=   ÂÞš@Xpþ`îÇÁ¥AfêCàl¯‡X´A
'¤R4£"¿ª-VUUÌ˜ UDJÝ jh ¼¯ „ïë(¨Ø Täì üú6!•€¹äY‘-cßLQ'
ùá¼œ5L-eŒòøð\§  |dHŸô  …àPïo ½Q\J™d3ùdÎ"Kî}†¡Ç>¯ Êk,^  Ü¢TVŸ^+J¥¢ŒEƒYN
vo¾©;»·ë+] †-©{° çø^VV{*&äëNß ¾5]ô°ýèjŸ• ± ã$è2-lfOÊY6Ëú”¯  sÐÊ«q¸@•9-
èâ“ì›’ÞeeŠñ  Ë#:- lµC6¼Ü   ¢ÝRƒçÀ£².i :að2 ¥•ïÜÔµhó•Þ
,g2V}ÁX|ý¼§©B^çh„èW}Ø*ËK †wGØ™±üY·|÷ @¡é9³€è[Bé•s¯^Z¾  °îe³²—rr-
hB&¯ªTfÏédYLç b úµçù t™&éíþña»z¢UZð  èãe”xáßy¢•ø¯[¼QMËéÎ®þ ~éáj
endstream
endobj
11 0 obj
1136
endobj
14 0 obj
<<
/Type /XObject
/Subtype /Image
/Name /im1
/Filter /DCTDecode
/Width 811
/Height 283
```

```
/BitsPerComponent 8
/ColorSpace /DeviceGray
/Length 15 0 R
>>
...
/Length 25 0 R
/Filter /FlateDecode
>>
stream
H‰}SËr›0 ý þá.“™XE @t—ÄöŒ Mv³òF ÅQÆ  •ûøú^!ŒñL› 3B°: «s¿Bô°‰DÆ g@…$I
°™G7p
›·hÆ A Ì¨$ óðaÉ•Æ$–ø÷ Íp%bŠë bBÓ~ù n–fß5
hÒ£ J  q•?>×²P      •(!L   “§#f’˜sã Ö™–  ~ÜÂ³5…þÒØ_¿•‵¸‡¡”yÿ¡
–8 D¢¥œ¤r°D•R4Ò[ ÊøDÑek´•") ) "O Å“v–,Ú  èV  XpÆµ°RSµÊÔ% ¸ }PpÝ®4'SšÒ,®
´Æù½á ƒ  ¦n–»ƒ î• g¨•§WÖ²¸w†þäØÊTŒ  ¡¸mÔN ^Uåé[¿u^G… ¦•io äGâ–8
ëÐí •1IÄä¡Xœñ >–ý ñõA» çzŸ”wŸƒæœäU'áÄ¿‡˜¼G9˜ 7·,Í†Ô`Ñ{è¹ÿ /Á Q
”Òsg$“CWÏŸW ØÞ¿?WëÍâÓ=Ì °zÚ,¾–ï ÛÛ`
'€‰    p  x_¢•ËàS Ó° †X: š&,Îùy¤àÃ @£qjç³sÒ8êƒ
Ì‡nÔe4[aô:×ùªÆX|ØF <¡î<Ø²ÏÿuX(îö„8EŒæ• £ŠdI, NƒF~ÌÌ«.L¥}nJ'ûl–.ª´wXÓªjçÙª
¯V'½ŸÈ+¯´Õ
V^•–[ °ìêÂØZ5þø:í×M]l¢¿ÄA.ƒ
endstream
endobj
25 0 obj
589
endobj
...
/Size 38
/Root 3 0 R
/Info 1 0 R
/ID [<f789d79e12493517c45121814172ee25><f789d79e12493517c45121814172ee25>]
>>
startxref
105320
%%EOF
```

2o. Fragmento – (PDF) Script de Áudio gerado pelo tVoice.

```
\Vce=Speaker=Cláudia\ Autor: Teste. \Rst\ \Vce=Speaker=Cláudia\ Palavras
Chave: Acessibilidade, Sistemas distribuídos, Síntese e Reconhecimento de
voz \Rst\ 4. O Sistema VoiceProxy.
```

```
O projeto VoiceProxy tem como principal meta, desde sua primeira versão,
desenvolvida em 2001, promover a Acessibilidade dentro do ambiente Web
[16]. Tentando minimizar as barreiras encontradas pelos deficientes
visuais no acesso às informações disponibilizadas através de documentos
HTML, o sistema utiliza a síntese e o reconhecimento de voz como
método de acesso para os portadores de necessidades especiais, mais
especificamente os deficientes visuais. Em sua primeira versão,
resumidamente descrevendo sua funcionalidade, o sistema funcionava
instalado na máquina do usuário, semelhantemente a um browser, que depois
de configurado e posto em operação, aguardava por um comando do usuário
solicitando-lhe acesso a uma determinada página Web, seja pela fala
(reconhecimento de voz), através de um microfone, ou teclado. Em seguida o
VoiceProxy buscava a página solicitada, realizava a extração do conteúdo
textual da mesma e o lia (síntese de voz) para o usuário [16]. A figura 14
nos apresenta a disposição da primeira versão do sistema VoiceProxy em
relação ao usuário e à Web.
...
A proposta do VoiceProxy é tentar fazer com que o maior número de usuários
especiais possam utilizar a Internet, sejam eles possuidores de maior ou
menor capacitação para o uso da máquina, e assim, fazer com que eles
possam ter na Internet uma fonte de informação crescente e inesgotável.
Assim como na primeira versão, o acesso dos portadores de necessidades
especiais aos documentos, utilizados na extração de informação, ainda
sofre restrições diretamente relacionadas aos tipos de documentos
utilizados no processamento. Tais restrições e limitações, por estarem
diretamente ligadas ao subsistema de tradução, parte integrante do sistema
VoiceProxy, serão melhor apresentadas no capítulo seguinte. \Rst\
\xPfl=70\ Fim do Documento.
```

3o. Fragmento – Documento XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>: : : SUPER Online : : : SUPERnotícias : : :</title>
</head>
<body link="#008080" vlink="#008080" alink="#808000" bgcolor="#999966">
<table border="1" width="100%" height="94">
  <tr> <td width="100%" height="63"></td></tr>
  <tr> <td width="100%" height="19"></td></tr>
</table>
<hr>
<div align="center">
  <center> <table border="1" width="550" bordercolor="#FFFFFF"
bgcolor="#008080">
    <tr> <td width="34%" align="center"><font face="Tahoma" size="2"><b>
23/04/2002</b></font></td>
         <td width="33%" align="center"><font face="Tahoma" size="2"><b>
16:23</b></font></td>
         <td width="33%" align="center"><font face="Tahoma" size="2"><b>
TECNOLOGIA</b></font></td>
    </tr>
  </table></center>
</div>
<div align="center">
  <center> <table border="2" width="550" bgcolor="#C0C0C0"
cellpadding="10" bordercolor="#FFFFFF" height="432">
    <tr> <td width="410" rowspan="10" height="408"> <P class=titulo
align="center"><b><font face="Tahoma" size="5"><br> Display tátil mostra
imagens para cegos</font></b></P>
  <center> <P class=texto align="justify" style="margin-left: 7; margin-
right: 7"><font face="Tahoma">
        Usuários cegos agora podem navegar pela internet sem se frustrar
por não ver o design arrojado desenvolvido na web. Um display que reproduz
em relevo as informações gráficas que aparecem na tela de um PC permite
que as imagens possam ser &quot;lidas&quot; pelo tato. Instalado no início
deste mês na Biblioteca Central de Osaka, no Japão, o painel é composto
```

```
por 3 072 pinos de plástico com 1,6 mm de diâmetro cada. Segundo o jornal
<I>The Japan Times</I>, o dispositivo de 14 por 16 cm, que custou 5
milhões de ienes (cerca de 90 mil reais), foi desenvolvido por uma empresa
privada em parceria com a Nasda – Agência Nacional de Desenvolvimento
Espacial do Japão – para que um pesquisador cego pudesse controlar um
sistema de monitoramento de satélites. - <A class=link_sign
href="mailto:gyamaguchi@abril.com.br">por Gabriela Yamaguchi<br><br>
</A></font></P>
</center>
    <tr> <td width="140" align="center" height="20"> <A class=link_red
href="http://superinteressante.abril.uol.com.br/aberta/noticias/index_uni.
html"><b><font face="Tahoma" size="1">UNIVERSO</font></b></A></td>
         <td width="140" align="center" height="20"> <A class=link_red
href="http://superinteressante.abril.uol.com.br/aberta/noticias/index_eco.
html"><b><font face="Tahoma" size="1">ECOLOGIA</font></b></A></td>
    </tr>
    <tr> <td width="140" align="center" height="18"> <A class=link_red
href="http://superinteressante.abril.uol.com.br/aberta/noticias/index_his.
html"><b><font face="Tahoma" size="1">HISTÓRIA</font></b></A></td>
    </tr>
...
</body>
</html>
```

3o. Fragmento - Script de Áudio gerado pelo tVoice.

```
\Rst\ \xPfl=70\ Título da página: \Pau=200\ : : : SUPER Online : : :
SUPERnotícias : : : \Rst\ \xPfl=70\ \Pau=200\ 23/04/2002 \Rst\ \xPfl=70\
\Pau=200\ 16:23 \Rst\ \xPfl=70\ \Pau=200\ TECNOLOGIA \Rst\ \Pau=200\
Display tátil mostra imagens para cegos \Rst\ \Pau=200\       Usuários
cegos agora podem navegar pela internet sem se frustrar por não ver o
design arrojado desenvolvido na web. Um display que reproduz em relevo as
informações gráficas que aparecem na tela de um PC permite que as imagens
possam ser "lidas" pelo tato. Instalado no início deste mês na Biblioteca
Central de Osaka, no Japão, o painel é composto por 3 072 pinos de
plástico com 1,6 mm de diâmetro cada. Segundo o jornal \Rst\ \xPfl=70\
\Pau=200\ The Japan Times \Rst\ \Pau=200\ , o dispositivo de 14 por 16 cm,
que custou 5 milhões de ienes (cerca de 90 mil reais), foi desenvolvido
por uma empresa privada em parceria com a Nasda – Agência Nacional de
Desenvolvimento Espacial do Japão – para que um pesquisador cego pudesse
```

```
controlar um sistema de monitoramento de satélites. - \Rst\
\Vce=Speaker=Cláudia\ Link \Pau=200\ por Gabriela Yamaguchi \Rst\
\Vce=Speaker=Cláudia\ Link \xPfl=70\ \Pau=200\ UNIVERSO \Rst\ \xPfl=70\
\Vce=Speaker=Cláudia\ Link \xPfl=70\ \Pau=200\ ECOLOGIA \Rst\
\Vce=Speaker=Cláudia\ Link \xPfl=70\ \Pau=200\ HISTÓRIA \Rst\ \Pau=200\
...
\Rst\
\xPfl=70\ Fim do Documento.
```

ANEXO II

Fragmentos de resultados obtidos com testes GET Linux

Máquina **Caetano** (10.9.98.17)	
Arquivo **rst_Caetano1**	Arquivo **rst_Caetano2**
Wed Dec 29 17:13:00 BRST 2004	Wed Dec 29 17:13:01 BRST 2004
Wed Dec 29 17:13:01 BRST 2004	Wed Dec 29 17:13:03 BRST 2004
Wed Dec 29 17:14:00 BRST 2004	Wed Dec 29 17:14:01 BRST 2004
Wed Dec 29 17:14:01 BRST 2004	Wed Dec 29 17:14:05 BRST 2004
Wed Dec 29 17:15:00 BRST 2004	Wed Dec 29 17:15:01 BRST 2004
Wed Dec 29 17:15:01 BRST 2004	Wed Dec 29 17:15:03 BRST 2004
Wed Dec 29 17:16:00 BRST 2004	Wed Dec 29 17:16:01 BRST 2004
Wed Dec 29 17:16:00 BRST 2004	Wed Dec 29 17:16:04 BRST 2004
Wed Dec 29 17:16:59 BRST 2004	Wed Dec 29 17:17:00 BRST 2004
Wed Dec 29 17:17:00 BRST 2004	Wed Dec 29 17:17:02 BRST 2004
Wed Dec 29 17:18:00 BRST 2004	Wed Dec 29 17:18:01 BRST 2004
Wed Dec 29 17:18:01 BRST 2004	Wed Dec 29 17:18:04 BRST 2004
Wed Dec 29 17:19:00 BRST 2004	Wed Dec 29 17:19:01 BRST 2004
Wed Dec 29 17:19:00 BRST 2004	Wed Dec 29 17:19:02 BRST 2004
Wed Dec 29 17:19:59 BRST 2004	Wed Dec 29 17:20:00 BRST 2004
Wed Dec 29 17:20:00 BRST 2004	Wed Dec 29 17:20:02 BRST 2004
Wed Dec 29 17:21:00 BRST 2004	Wed Dec 29 17:21:01 BRST 2004
Wed Dec 29 17:21:01 BRST 2004	Wed Dec 29 17:21:03 BRST 2004
Wed Dec 29 17:22:00 BRST 2004	Wed Dec 29 17:22:01 BRST 2004
Wed Dec 29 17:22:00 BRST 2004	Wed Dec 29 17:22:03 BRST 2004
Wed Dec 29 17:22:59 BRST 2004	Wed Dec 29 17:23:00 BRST 2004
Wed Dec 29 17:23:00 BRST 2004	Wed Dec 29 17:23:01 BRST 2004
Wed Dec 29 17:24:00 BRST 2004	Wed Dec 29 17:24:01 BRST 2004
Wed Dec 29 17:24:01 BRST 2004	Wed Dec 29 17:24:02 BRST 2004
Wed Dec 29 17:25:00 BRST 2004	Wed Dec 29 17:25:00 BRST 2004
Wed Dec 29 17:25:01 BRST 2004	Wed Dec 29 17:25:02 BRST 2004
Wed Dec 29 17:26:00 BRST 2004	Wed Dec 29 17:26:01 BRST 2004
Wed Dec 29 17:26:00 BRST 2004	Wed Dec 29 17:26:03 BRST 2004
Wed Dec 29 17:26:59 BRST 2004	Wed Dec 29 17:27:00 BRST 2004
Wed Dec 29 17:27:00 BRST 2004	Wed Dec 29 17:27:04 BRST 2004
Wed Dec 29 17:28:00 BRST 2004	Wed Dec 29 17:28:01 BRST 2004
Wed Dec 29 17:28:01 BRST 2004	Wed Dec 29 17:28:03 BRST 2004
Wed Dec 29 17:29:00 BRST 2004	Wed Dec 29 17:29:01 BRST 2004
Wed Dec 29 17:29:01 BRST 2004	Wed Dec 29 17:29:02 BRST 2004
Wed Dec 29 17:30:00 BRST 2004	Wed Dec 29 17:30:01 BRST 2004
Wed Dec 29 17:30:01 BRST 2004	Wed Dec 29 17:30:03 BRST 2004
Wed Dec 29 17:31:00 BRST 2004	Wed Dec 29 17:31:01 BRST 2004
Wed Dec 29 17:31:00 BRST 2004	Wed Dec 29 17:31:04 BRST 2004
Wed Dec 29 17:31:59 BRST 2004	Wed Dec 29 17:32:00 BRST 2004

Máquina **Leao** (10.258.0.16)	
Arquivo **rst_Leao1**	Arquivo **rst_Leao2**
Wed Dec 29 16:32:00 BRT 2004 Wed Dec 29 16:32:04 BRT 2004 Wed Dec 29 16:33:00 BRT 2004 Wed Dec 29 16:33:04 BRT 2004 Wed Dec 29 16:34:00 BRT 2004 Wed Dec 29 16:34:05 BRT 2004 Wed Dec 29 16:35:00 BRT 2004 Wed Dec 29 16:35:05 BRT 2004 Wed Dec 29 16:36:00 BRT 2004 Wed Dec 29 16:36:04 BRT 2004 Wed Dec 29 16:37:00 BRT 2004 Wed Dec 29 16:37:04 BRT 2004 Wed Dec 29 16:38:00 BRT 2004 Wed Dec 29 16:38:05 BRT 2004 Wed Dec 29 16:39:01 BRT 2004 Wed Dec 29 16:39:05 BRT 2004 Wed Dec 29 16:40:00 BRT 2004 Wed Dec 29 16:40:06 BRT 2004 Wed Dec 29 16:42:00 BRT 2004 Wed Dec 29 16:42:05 BRT 2004 Wed Dec 29 16:43:00 BRT 2004 Wed Dec 29 16:43:05 BRT 2004 Wed Dec 29 16:44:00 BRT 2004 Wed Dec 29 16:44:05 BRT 2004 Wed Dec 29 16:46:00 BRT 2004 Wed Dec 29 16:46:05 BRT 2004 Wed Dec 29 16:47:01 BRT 2004 Wed Dec 29 16:47:05 BRT 2004 Wed Dec 29 16:48:01 BRT 2004 Wed Dec 29 16:48:06 BRT 2004 Wed Dec 29 16:49:01 BRT 2004 Wed Dec 29 16:49:06 BRT 2004 Wed Dec 29 16:50:00 BRT 2004 Wed Dec 29 16:50:05 BRT 2004 Wed Dec 29 16:51:00 BRT 2004 Wed Dec 29 16:51:04 BRT 2004 Wed Dec 29 16:52:01 BRT 2004 Wed Dec 29 16:52:06 BRT 2004 Wed Dec 29 16:53:00 BRT 2004 Wed Dec 29 16:53:05 BRT 2004 Wed Dec 29 16:54:00 BRT 2004 Wed Dec 29 16:54:05 BRT 2004 Wed Dec 29 16:55:01 BRT 2004 Wed Dec 29 16:55:06 BRT 2004 Wed Dec 29 16:55:59 BRT 2004 Wed Dec 29 16:56:03 BRT 2004	Wed Dec 29 16:34:00 BRT 2004 Wed Dec 29 16:34:05 BRT 2004 Wed Dec 29 16:35:00 BRT 2004 Wed Dec 29 16:35:06 BRT 2004 Wed Dec 29 16:36:00 BRT 2004 Wed Dec 29 16:36:05 BRT 2004 Wed Dec 29 16:36:59 BRT 2004 Wed Dec 29 16:37:04 BRT 2004 Wed Dec 29 16:38:01 BRT 2004 Wed Dec 29 16:38:06 BRT 2004 Wed Dec 29 16:39:01 BRT 2004 Wed Dec 29 16:39:06 BRT 2004 Wed Dec 29 16:40:00 BRT 2004 Wed Dec 29 16:40:06 BRT 2004 Wed Dec 29 16:41:00 BRT 2004 Wed Dec 29 16:41:07 BRT 2004 Wed Dec 29 16:42:00 BRT 2004 Wed Dec 29 16:42:06 BRT 2004 Wed Dec 29 16:43:00 BRT 2004 Wed Dec 29 16:43:05 BRT 2004 Wed Dec 29 16:44:00 BRT 2004 Wed Dec 29 16:44:06 BRT 2004 Wed Dec 29 16:45:01 BRT 2004 Wed Dec 29 16:45:07 BRT 2004 Wed Dec 29 16:46:00 BRT 2004 Wed Dec 29 16:46:05 BRT 2004 Wed Dec 29 16:47:00 BRT 2004 Wed Dec 29 16:47:05 BRT 2004 Wed Dec 29 16:48:00 BRT 2004 Wed Dec 29 16:48:06 BRT 2004 Wed Dec 29 16:49:01 BRT 2004 Wed Dec 29 16:49:06 BRT 2004 Wed Dec 29 16:50:01 BRT 2004 Wed Dec 29 16:50:06 BRT 2004 Wed Dec 29 16:51:00 BRT 2004 Wed Dec 29 16:51:06 BRT 2004 Wed Dec 29 16:52:01 BRT 2004 Wed Dec 29 16:52:06 BRT 2004 Wed Dec 29 16:53:00 BRT 2004 Wed Dec 29 16:53:05 BRT 2004 Wed Dec 29 16:54:00 BRT 2004 Wed Dec 29 16:54:06 BRT 2004 Wed Dec 29 16:55:01 BRT 2004 Wed Dec 29 16:55:07 BRT 2004 Wed Dec 29 16:55:59 BRT 2004 Wed Dec 29 16:56:04 BRT 2004

Máquina **Cazuza** (10.9.98.18)	
Arquivo **rst_Cazuza1**	Arquivo **rst_Cazuza2**
Wed Dec 29 17:37:00 BRST 2004 Wed Dec 29 17:37:02 BRST 2004 Wed Dec 29 17:38:00 BRST 2004 Wed Dec 29 17:38:01 BRST 2004 Wed Dec 29 17:39:00 BRST 2004 Wed Dec 29 17:39:02 BRST 2004 Wed Dec 29 17:40:00 BRST 2004 Wed Dec 29 17:40:01 BRST 2004 Wed Dec 29 17:41:00 BRST 2004 Wed Dec 29 17:41:02 BRST 2004 Wed Dec 29 17:42:01 BRST 2004 Wed Dec 29 17:42:04 BRST 2004 Wed Dec 29 17:43:00 BRST 2004 Wed Dec 29 17:43:03 BRST 2004 Wed Dec 29 17:44:00 BRST 2004 Wed Dec 29 17:44:05 BRST 2004 Wed Dec 29 17:45:00 BRST 2004 Wed Dec 29 17:45:06 BRST 2004 Wed Dec 29 17:46:00 BRST 2004 Wed Dec 29 17:46:02 BRST 2004 Wed Dec 29 17:47:00 BRST 2004 Wed Dec 29 17:47:02 BRST 2004 Wed Dec 29 17:48:00 BRST 2004 Wed Dec 29 17:48:02 BRST 2004 Wed Dec 29 17:49:01 BRST 2004 Wed Dec 29 17:49:03 BRST 2004 Wed Dec 29 17:50:00 BRST 2004 Wed Dec 29 17:50:01 BRST 2004 Wed Dec 29 17:51:00 BRST 2004 Wed Dec 29 17:51:01 BRST 2004 Wed Dec 29 17:52:00 BRST 2004 Wed Dec 29 17:52:01 BRST 2004 Wed Dec 29 17:53:00 BRST 2004 Wed Dec 29 17:53:01 BRST 2004 Wed Dec 29 17:54:01 BRST 2004 Wed Dec 29 17:54:02 BRST 2004 Wed Dec 29 17:54:59 BRST 2004 Wed Dec 29 17:55:01 BRST 2004 Wed Dec 29 17:56:01 BRST 2004 Wed Dec 29 17:56:04 BRST 2004 Wed Dec 29 17:57:00 BRST 2004 Wed Dec 29 17:57:02 BRST 2004 Wed Dec 29 17:58:01 BRST 2004 Wed Dec 29 17:58:02 BRST 2004 Wed Dec 29 17:59:00 BRST 2004 Wed Dec 29 17:59:01 BRST 2004	Wed Dec 29 17:37:01 BRST 2004 Wed Dec 29 17:37:05 BRST 2004 Wed Dec 29 17:38:00 BRST 2004 Wed Dec 29 17:38:02 BRST 2004 Wed Dec 29 17:39:00 BRST 2004 Wed Dec 29 17:39:03 BRST 2004 Wed Dec 29 17:40:00 BRST 2004 Wed Dec 29 17:40:01 BRST 2004 Wed Dec 29 17:41:00 BRST 2004 Wed Dec 29 17:41:02 BRST 2004 Wed Dec 29 17:42:01 BRST 2004 Wed Dec 29 17:42:04 BRST 2004 Wed Dec 29 17:43:00 BRST 2004 Wed Dec 29 17:43:03 BRST 2004 Wed Dec 29 17:44:00 BRST 2004 Wed Dec 29 17:44:06 BRST 2004 Wed Dec 29 17:45:00 BRST 2004 Wed Dec 29 17:45:06 BRST 2004 Wed Dec 29 17:46:00 BRST 2004 Wed Dec 29 17:46:02 BRST 2004 Wed Dec 29 17:47:00 BRST 2004 Wed Dec 29 17:47:02 BRST 2004 Wed Dec 29 17:48:00 BRST 2004 Wed Dec 29 17:48:02 BRST 2004 Wed Dec 29 17:49:01 BRST 2004 Wed Dec 29 17:49:03 BRST 2004 Wed Dec 29 17:50:00 BRST 2004 Wed Dec 29 17:50:02 BRST 2004 Wed Dec 29 17:51:00 BRST 2004 Wed Dec 29 17:51:02 BRST 2004 Wed Dec 29 17:52:00 BRST 2004 Wed Dec 29 17:52:02 BRST 2004 Wed Dec 29 17:53:00 BRST 2004 Wed Dec 29 17:53:02 BRST 2004 Wed Dec 29 17:54:01 BRST 2004 Wed Dec 29 17:54:02 BRST 2004 Wed Dec 29 17:54:59 BRST 2004 Wed Dec 29 17:55:01 BRST 2004 Wed Dec 29 17:56:01 BRST 2004 Wed Dec 29 17:56:04 BRST 2004 Wed Dec 29 17:57:00 BRST 2004 Wed Dec 29 17:57:02 BRST 2004 Wed Dec 29 17:58:01 BRST 2004 Wed Dec 29 17:58:03 BRST 2004 Wed Dec 29 17:59:00 BRST 2004 Wed Dec 29 17:59:01 BRST 2004

Máquina **Elba** (10.9.98.26)	
Arquivo **rst_Elba1**	Arquivo **rst_Elba2**
Wed Dec 29 12:46:00 BRST 2004 Wed Dec 29 12:46:02 BRST 2004 Wed Dec 29 12:47:00 BRST 2004 Wed Dec 29 12:47:04 BRST 2004 Wed Dec 29 12:48:00 BRST 2004 Wed Dec 29 12:48:03 BRST 2004 Wed Dec 29 12:49:00 BRST 2004 Wed Dec 29 12:49:03 BRST 2004 Wed Dec 29 12:50:01 BRST 2004 Wed Dec 29 12:50:02 BRST 2004 Wed Dec 29 12:51:00 BRST 2004 Wed Dec 29 12:51:01 BRST 2004 Wed Dec 29 12:52:00 BRST 2004 Wed Dec 29 12:52:02 BRST 2004 Wed Dec 29 12:53:00 BRST 2004 Wed Dec 29 12:53:02 BRST 2004 Wed Dec 29 12:54:00 BRST 2004 Wed Dec 29 12:54:02 BRST 2004 Wed Dec 29 12:55:00 BRST 2004 Wed Dec 29 12:55:02 BRST 2004 Wed Dec 29 12:56:01 BRST 2004 Wed Dec 29 12:56:02 BRST 2004 Wed Dec 29 12:57:01 BRST 2004 Wed Dec 29 12:57:02 BRST 2004 Wed Dec 29 12:58:00 BRST 2004 Wed Dec 29 12:58:02 BRST 2004 Wed Dec 29 12:59:00 BRST 2004 Wed Dec 29 12:59:02 BRST 2004 Wed Dec 29 13:00:00 BRST 2004 Wed Dec 29 13:00:03 BRST 2004 Wed Dec 29 13:01:01 BRST 2004 Wed Dec 29 13:01:02 BRST 2004 Wed Dec 29 13:02:00 BRST 2004 Wed Dec 29 13:02:02 BRST 2004 Wed Dec 29 13:03:00 BRST 2004 Wed Dec 29 13:03:02 BRST 2004 Wed Dec 29 13:04:00 BRST 2004 Wed Dec 29 13:04:02 BRST 2004 Wed Dec 29 13:05:01 BRST 2004 Wed Dec 29 13:05:02 BRST 2004 Wed Dec 29 13:05:59 BRST 2004 Wed Dec 29 13:06:01 BRST 2004 Wed Dec 29 13:07:01 BRST 2004 Wed Dec 29 13:07:02 BRST 2004 Wed Dec 29 13:08:00 BRST 2004 Wed Dec 29 13:08:02 BRST 2004	Wed Dec 29 12:46:00 BRST 2004 Wed Dec 29 12:46:02 BRST 2004 Wed Dec 29 12:47:00 BRST 2004 Wed Dec 29 12:47:05 BRST 2004 Wed Dec 29 12:48:00 BRST 2004 Wed Dec 29 12:48:03 BRST 2004 Wed Dec 29 12:49:00 BRST 2004 Wed Dec 29 12:49:02 BRST 2004 Wed Dec 29 12:50:01 BRST 2004 Wed Dec 29 12:50:03 BRST 2004 Wed Dec 29 12:51:00 BRST 2004 Wed Dec 29 12:51:02 BRST 2004 Wed Dec 29 12:52:00 BRST 2004 Wed Dec 29 12:52:02 BRST 2004 Wed Dec 29 12:53:00 BRST 2004 Wed Dec 29 12:53:02 BRST 2004 Wed Dec 29 12:54:00 BRST 2004 Wed Dec 29 12:54:02 BRST 2004 Wed Dec 29 12:55:00 BRST 2004 Wed Dec 29 12:55:03 BRST 2004 Wed Dec 29 12:56:01 BRST 2004 Wed Dec 29 12:56:04 BRST 2004 Wed Dec 29 12:57:00 BRST 2004 Wed Dec 29 12:57:03 BRST 2004 Wed Dec 29 12:58:00 BRST 2004 Wed Dec 29 12:58:03 BRST 2004 Wed Dec 29 12:59:00 BRST 2004 Wed Dec 29 12:59:02 BRST 2004 Wed Dec 29 13:00:00 BRST 2004 Wed Dec 29 13:00:02 BRST 2004 Wed Dec 29 13:01:01 BRST 2004 Wed Dec 29 13:01:03 BRST 2004 Wed Dec 29 13:02:00 BRST 2004 Wed Dec 29 13:02:02 BRST 2004 Wed Dec 29 13:03:00 BRST 2004 Wed Dec 29 13:03:02 BRST 2004 Wed Dec 29 13:04:00 BRST 2004 Wed Dec 29 13:04:03 BRST 2004 Wed Dec 29 13:05:01 BRST 2004 Wed Dec 29 13:05:02 BRST 2004 Wed Dec 29 13:05:59 BRST 2004 Wed Dec 29 13:06:02 BRST 2004 Wed Dec 29 13:07:01 BRST 2004 Wed Dec 29 13:07:02 BRST 2004 Wed Dec 29 13:08:00 BRST 2004 Wed Dec 29 13:08:02 BRST 2004

Máquina **Gonzagao** (10.9.98.15)	
Arquivo **rst_Gonzagao1**	Arquivo **rst_Gonzagao2**
Wed Dec 29 17:12:00 BRST 2004 Wed Dec 29 17:12:02 BRST 2004 Wed Dec 29 17:13:00 BRST 2004 Wed Dec 29 17:13:01 BRST 2004 Wed Dec 29 17:14:00 BRST 2004 Wed Dec 29 17:14:01 BRST 2004 Wed Dec 29 17:15:01 BRST 2004 Wed Dec 29 17:15:02 BRST 2004 Wed Dec 29 17:16:02 BRST 2004 Wed Dec 29 17:16:02 BRST 2004 Wed Dec 29 17:17:00 BRST 2004 Wed Dec 29 17:17:01 BRST 2004 Wed Dec 29 17:18:00 BRST 2004 Wed Dec 29 17:18:01 BRST 2004 Wed Dec 29 17:19:00 BRST 2004 Wed Dec 29 17:19:01 BRST 2004 Wed Dec 29 17:20:01 BRST 2004 Wed Dec 29 17:20:02 BRST 2004 Wed Dec 29 17:21:01 BRST 2004 Wed Dec 29 17:21:02 BRST 2004 Wed Dec 29 17:22:00 BRST 2004 Wed Dec 29 17:22:01 BRST 2004 Wed Dec 29 17:23:01 BRST 2004 Wed Dec 29 17:23:02 BRST 2004 Wed Dec 29 17:24:00 BRST 2004 Wed Dec 29 17:24:00 BRST 2004 Wed Dec 29 17:25:01 BRST 2004 Wed Dec 29 17:25:02 BRST 2004 Wed Dec 29 17:26:02 BRST 2004 Wed Dec 29 17:26:02 BRST 2004 Wed Dec 29 17:27:00 BRST 2004 Wed Dec 29 17:27:01 BRST 2004 Wed Dec 29 17:28:00 BRST 2004 Wed Dec 29 17:28:01 BRST 2004	Wed Dec 29 17:12:01 BRST 2004 Wed Dec 29 17:12:04 BRST 2004 Wed Dec 29 17:13:02 BRST 2004 Wed Dec 29 17:13:05 BRST 2004 Wed Dec 29 17:14:01 BRST 2004 Wed Dec 29 17:14:03 BRST 2004 Wed Dec 29 17:15:01 BRST 2004 Wed Dec 29 17:15:02 BRST 2004 Wed Dec 29 17:16:02 BRST 2004 Wed Dec 29 17:16:04 BRST 2004 Wed Dec 29 17:17:01 BRST 2004 Wed Dec 29 17:17:03 BRST 2004 Wed Dec 29 17:18:00 BRST 2004 Wed Dec 29 17:18:02 BRST 2004 Wed Dec 29 17:19:01 BRST 2004 Wed Dec 29 17:19:04 BRST 2004 Wed Dec 29 17:20:01 BRST 2004 Wed Dec 29 17:20:03 BRST 2004 Wed Dec 29 17:21:01 BRST 2004 Wed Dec 29 17:21:04 BRST 2004 Wed Dec 29 17:22:01 BRST 2004 Wed Dec 29 17:22:02 BRST 2004 Wed Dec 29 17:23:02 BRST 2004 Wed Dec 29 17:23:04 BRST 2004 Wed Dec 29 17:24:01 BRST 2004 Wed Dec 29 17:24:02 BRST 2004 Wed Dec 29 17:25:01 BRST 2004 Wed Dec 29 17:25:03 BRST 2004 Wed Dec 29 17:26:02 BRST 2004 Wed Dec 29 17:26:03 BRST 2004 Wed Dec 29 17:27:01 BRST 2004 Wed Dec 29 17:27:03 BRST 2004 Wed Dec 29 17:28:01 BRST 2004 Wed Dec 29 17:28:03 BRST 2004

Máquina **Jobim** (10.9.98.16)	
Arquivo **rst_Jobim1**	Arquivo **rst_Jobim2**
Wed Dec 29 17:12:01 BRST 2004 Wed Dec 29 17:12:02 BRST 2004 Wed Dec 29 17:13:00 BRST 2004 Wed Dec 29 17:13:01 BRST 2004 Wed Dec 29 18:14:00 BRST 2004 Wed Dec 29 18:14:02 BRST 2004 Wed Dec 29 18:15:00 BRST 2004 Wed Dec 29 18:15:02 BRST 2004 Wed Dec 29 18:16:01 BRST 2004 Wed Dec 29 18:16:02 BRST 2004 Wed Dec 29 18:17:01 BRST 2004 Wed Dec 29 18:17:04 BRST 2004 Wed Dec 29 18:18:00 BRST 2004 Wed Dec 29 18:18:02 BRST 2004 Wed Dec 29 18:19:00 BRST 2004 Wed Dec 29 18:19:03 BRST 2004 Wed Dec 29 18:20:01 BRST 2004 Wed Dec 29 18:20:04 BRST 2004 Wed Dec 29 18:21:00 BRST 2004 Wed Dec 29 18:21:03 BRST 2004 Wed Dec 29 18:22:01 BRST 2004 Wed Dec 29 18:22:04 BRST 2004 Wed Dec 29 18:23:17 BRST 2004 Wed Dec 29 18:23:19 BRST 2004 Wed Dec 29 18:24:00 BRST 2004 Wed Dec 29 18:24:02 BRST 2004 Wed Dec 29 18:25:01 BRST 2004 Wed Dec 29 18:25:02 BRST 2004 Wed Dec 29 18:26:01 BRST 2004 Wed Dec 29 18:26:02 BRST 2004 Wed Dec 29 18:27:03 BRST 2004 Wed Dec 29 18:27:04 BRST 2004 Wed Dec 29 18:28:00 BRST 2004 Wed Dec 29 18:28:01 BRST 2004 Wed Dec 29 18:29:00 BRST 2004 Wed Dec 29 18:29:01 BRST 2004 Wed Dec 29 18:30:01 BRST 2004 Wed Dec 29 18:30:02 BRST 2004 Wed Dec 29 18:31:00 BRST 2004 Wed Dec 29 18:31:01 BRST 2004 Wed Dec 29 18:32:00 BRST 2004 Wed Dec 29 18:32:01 BRST 2004 Wed Dec 29 18:33:00 BRST 2004 Wed Dec 29 18:33:01 BRST 2004 Wed Dec 29 18:34:00 BRST 2004 Wed Dec 29 18:34:00 BRST 2004	Wed Dec 29 17:12:02 BRST 2004 Wed Dec 29 17:12:04 BRST 2004 Wed Dec 29 17:13:01 BRST 2004 Wed Dec 29 17:13:03 BRST 2004 Wed Dec 29 18:14:00 BRST 2004 Wed Dec 29 18:14:03 BRST 2004 Wed Dec 29 18:15:01 BRST 2004 Wed Dec 29 18:15:02 BRST 2004 Wed Dec 29 18:16:01 BRST 2004 Wed Dec 29 18:16:03 BRST 2004 Wed Dec 29 18:17:02 BRST 2004 Wed Dec 29 18:17:04 BRST 2004 Wed Dec 29 18:18:01 BRST 2004 Wed Dec 29 18:18:05 BRST 2004 Wed Dec 29 18:19:01 BRST 2004 Wed Dec 29 18:19:06 BRST 2004 Wed Dec 29 18:20:01 BRST 2004 Wed Dec 29 18:20:03 BRST 2004 Wed Dec 29 18:21:01 BRST 2004 Wed Dec 29 18:21:04 BRST 2004 Wed Dec 29 18:22:02 BRST 2004 Wed Dec 29 18:22:04 BRST 2004 Wed Dec 29 18:23:01 BRST 2004 Wed Dec 29 18:23:03 BRST 2004 Wed Dec 29 18:23:17 BRST 2004 Wed Dec 29 18:23:19 BRST 2004 Wed Dec 29 18:24:00 BRST 2004 Wed Dec 29 18:24:03 BRST 2004 Wed Dec 29 18:25:01 BRST 2004 Wed Dec 29 18:25:02 BRST 2004 Wed Dec 29 18:26:01 BRST 2004 Wed Dec 29 18:26:02 BRST 2004 Wed Dec 29 18:27:03 BRST 2004 Wed Dec 29 18:27:05 BRST 2004 Wed Dec 29 18:28:00 BRST 2004 Wed Dec 29 18:28:01 BRST 2004 Wed Dec 29 18:29:00 BRST 2004 Wed Dec 29 18:29:02 BRST 2004 Wed Dec 29 18:30:01 BRST 2004 Wed Dec 29 18:30:02 BRST 2004 Wed Dec 29 18:31:00 BRST 2004 Wed Dec 29 18:31:02 BRST 2004 Wed Dec 29 18:32:00 BRST 2004 Wed Dec 29 18:32:01 BRST 2004 Wed Dec 29 18:33:00 BRST 2004 Wed Dec 29 18:33:02 BRST 2004

Máquina **Oswaldo** (10.9.98.22)	
Arquivo **rst_Oswaldo1**	Arquivo **rst_Oswaldo2**
Wed Dec 29 17:00:00 BRST 2004	Wed Dec 29 17:00:01 BRST 2004
Wed Dec 29 17:00:01 BRST 2004	Wed Dec 29 17:00:03 BRST 2004
Wed Dec 29 17:01:00 BRST 2004	Wed Dec 29 17:01:01 BRST 2004
Wed Dec 29 17:01:03 BRST 2004	Wed Dec 29 17:01:04 BRST 2004
Wed Dec 29 17:02:00 BRST 2004	Wed Dec 29 17:02:01 BRST 2004
Wed Dec 29 17:02:01 BRST 2004	Wed Dec 29 17:02:05 BRST 2004
Wed Dec 29 17:03:00 BRST 2004	Wed Dec 29 17:03:01 BRST 2004
Wed Dec 29 17:03:02 BRST 2004	Wed Dec 29 17:03:03 BRST 2004
Wed Dec 29 17:04:01 BRST 2004	Wed Dec 29 17:04:01 BRST 2004
Wed Dec 29 17:04:02 BRST 2004	Wed Dec 29 17:04:03 BRST 2004
Wed Dec 29 17:05:00 BRST 2004	Wed Dec 29 17:05:00 BRST 2004
Wed Dec 29 17:05:02 BRST 2004	Wed Dec 29 17:05:02 BRST 2004
Wed Dec 29 17:06:01 BRST 2004	Wed Dec 29 17:06:01 BRST 2004
Wed Dec 29 17:06:02 BRST 2004	Wed Dec 29 17:06:03 BRST 2004
Wed Dec 29 17:07:00 BRST 2004	Wed Dec 29 17:07:01 BRST 2004
Wed Dec 29 17:07:02 BRST 2004	Wed Dec 29 17:07:02 BRST 2004
Wed Dec 29 17:08:00 BRST 2004	Wed Dec 29 17:08:00 BRST 2004
Wed Dec 29 17:08:02 BRST 2004	Wed Dec 29 17:08:04 BRST 2004
Wed Dec 29 17:09:01 BRST 2004	Wed Dec 29 17:09:01 BRST 2004
Wed Dec 29 17:09:02 BRST 2004	Wed Dec 29 17:09:03 BRST 2004
Wed Dec 29 17:09:59 BRST 2004	Wed Dec 29 17:09:59 BRST 2004
Wed Dec 29 17:10:00 BRST 2004	Wed Dec 29 17:10:01 BRST 2004
Wed Dec 29 17:11:00 BRST 2004	Wed Dec 29 17:11:00 BRST 2004
Wed Dec 29 17:11:03 BRST 2004	Wed Dec 29 17:11:05 BRST 2004
Wed Dec 29 17:12:00 BRST 2004	Wed Dec 29 17:12:01 BRST 2004
Wed Dec 29 17:12:02 BRST 2004	Wed Dec 29 17:12:03 BRST 2004
Wed Dec 29 17:13:00 BRST 2004	Wed Dec 29 17:13:00 BRST 2004
Wed Dec 29 17:13:01 BRST 2004	Wed Dec 29 17:13:04 BRST 2004
Wed Dec 29 17:14:00 BRST 2004	Wed Dec 29 17:14:00 BRST 2004
Wed Dec 29 17:14:02 BRST 2004	Wed Dec 29 17:14:03 BRST 2004
Wed Dec 29 17:15:01 BRST 2004	Wed Dec 29 17:15:01 BRST 2004
Wed Dec 29 17:15:03 BRST 2004	Wed Dec 29 17:15:05 BRST 2004
Wed Dec 29 17:16:01 BRST 2004	Wed Dec 29 17:16:01 BRST 2004
Wed Dec 29 17:16:02 BRST 2004	Wed Dec 29 17:16:03 BRST 2004
Wed Dec 29 17:17:00 BRST 2004	Wed Dec 29 17:17:01 BRST 2004
Wed Dec 29 17:17:00 BRST 2004	Wed Dec 29 17:17:02 BRST 2004
Wed Dec 29 17:18:00 BRST 2004	Wed Dec 29 17:18:00 BRST 2004
Wed Dec 29 17:18:02 BRST 2004	Wed Dec 29 17:18:03 BRST 2004
Wed Dec 29 17:19:01 BRST 2004	Wed Dec 29 17:19:01 BRST 2004
Wed Dec 29 17:19:02 BRST 2004	Wed Dec 29 17:19:03 BRST 2004
Wed Dec 29 17:20:00 BRST 2004	Wed Dec 29 17:20:00 BRST 2004
Wed Dec 29 17:20:02 BRST 2004	Wed Dec 29 17:20:04 BRST 2004
Wed Dec 29 17:21:00 BRST 2004	Wed Dec 29 17:21:01 BRST 2004
Wed Dec 29 17:21:03 BRST 2004	Wed Dec 29 17:21:03 BRST 2004
Wed Dec 29 17:22:00 BRST 2004	Wed Dec 29 17:22:00 BRST 2004
Wed Dec 29 17:22:01 BRST 2004	Wed Dec 29 17:22:02 BRST 2004

Máquina **Zeramalho** (10.9.98.27)	
Arquivo **rst_Zeramalho1**	Arquivo **rst_Zeramalho2**
Wed Dec 29 17:13:00 BRST 2004	Wed Dec 29 17:13:01 BRST 2004
Wed Dec 29 17:13:02 BRST 2004	Wed Dec 29 17:13:03 BRST 2004
Wed Dec 29 17:14:00 BRST 2004	Wed Dec 29 17:14:00 BRST 2004
Wed Dec 29 17:14:02 BRST 2004	Wed Dec 29 17:14:03 BRST 2004
Wed Dec 29 17:15:01 BRST 2004	Wed Dec 29 17:15:01 BRST 2004
Wed Dec 29 17:15:03 BRST 2004	Wed Dec 29 17:15:04 BRST 2004
Wed Dec 29 17:16:00 BRST 2004	Wed Dec 29 17:16:01 BRST 2004
Wed Dec 29 17:16:02 BRST 2004	Wed Dec 29 17:16:02 BRST 2004
Wed Dec 29 17:17:00 BRST 2004	Wed Dec 29 17:17:01 BRST 2004
Wed Dec 29 17:17:02 BRST 2004	Wed Dec 29 17:17:03 BRST 2004
Wed Dec 29 17:18:00 BRST 2004	Wed Dec 29 17:18:00 BRST 2004
Wed Dec 29 17:18:01 BRST 2004	Wed Dec 29 17:18:02 BRST 2004
Wed Dec 29 17:19:00 BRST 2004	Wed Dec 29 17:19:00 BRST 2004
Wed Dec 29 17:19:01 BRST 2004	Wed Dec 29 17:19:01 BRST 2004
Wed Dec 29 17:20:01 BRST 2004	Wed Dec 29 17:20:02 BRST 2004
Wed Dec 29 17:20:02 BRST 2004	Wed Dec 29 17:20:03 BRST 2004
Wed Dec 29 17:21:01 BRST 2004	Wed Dec 29 17:21:01 BRST 2004
Wed Dec 29 17:21:02 BRST 2004	Wed Dec 29 17:21:04 BRST 2004
Wed Dec 29 17:22:01 BRST 2004	Wed Dec 29 17:22:01 BRST 2004
Wed Dec 29 17:22:02 BRST 2004	Wed Dec 29 17:22:02 BRST 2004
Wed Dec 29 17:23:00 BRST 2004	Wed Dec 29 17:23:00 BRST 2004
Wed Dec 29 17:23:02 BRST 2004	Wed Dec 29 17:23:03 BRST 2004
Wed Dec 29 17:24:01 BRST 2004	Wed Dec 29 17:24:02 BRST 2004
Wed Dec 29 17:24:03 BRST 2004	Wed Dec 29 17:24:04 BRST 2004
Wed Dec 29 17:25:00 BRST 2004	Wed Dec 29 17:25:00 BRST 2004
Wed Dec 29 17:25:01 BRST 2004	Wed Dec 29 17:25:02 BRST 2004
Wed Dec 29 17:26:00 BRST 2004	Wed Dec 29 17:26:01 BRST 2004
Wed Dec 29 17:26:01 BRST 2004	Wed Dec 29 17:26:03 BRST 2004
Wed Dec 29 17:26:59 BRST 2004	Wed Dec 29 17:27:00 BRST 2004
Wed Dec 29 17:27:01 BRST 2004	Wed Dec 29 17:27:02 BRST 2004
Wed Dec 29 17:28:01 BRST 2004	Wed Dec 29 17:28:02 BRST 2004
Wed Dec 29 17:28:03 BRST 2004	Wed Dec 29 17:28:04 BRST 2004
Wed Dec 29 17:28:59 BRST 2004	Wed Dec 29 17:28:59 BRST 2004
Wed Dec 29 17:29:01 BRST 2004	Wed Dec 29 17:29:04 BRST 2004
Wed Dec 29 17:30:00 BRST 2004	Wed Dec 29 17:30:01 BRST 2004
Wed Dec 29 17:30:02 BRST 2004	Wed Dec 29 17:30:03 BRST 2004
Wed Dec 29 17:31:01 BRST 2004	Wed Dec 29 17:31:01 BRST 2004
Wed Dec 29 17:31:02 BRST 2004	Wed Dec 29 17:31:03 BRST 2004
Wed Dec 29 17:32:00 BRST 2004	Wed Dec 29 17:32:00 BRST 2004
Wed Dec 29 17:32:01 BRST 2004	Wed Dec 29 17:32:02 BRST 2004
Wed Dec 29 17:33:00 BRST 2004	Wed Dec 29 17:33:00 BRST 2004
Wed Dec 29 17:33:02 BRST 2004	Wed Dec 29 17:33:02 BRST 2004
Wed Dec 29 17:34:00 BRST 2004	Wed Dec 29 17:34:00 BRST 2004
Wed Dec 29 17:34:01 BRST 2004	Wed Dec 29 17:34:01 BRST 2004
Wed Dec 29 17:35:00 BRST 2004	Wed Dec 29 17:35:00 BRST 2004
Wed Dec 29 17:35:02 BRST 2004	Wed Dec 29 17:35:02 BRST 2004